Création automatisée de Scénarios de Formation pour l'enseignement d'activités métier dans un environnement informatique modulaire

Yohan Duval

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue par :
YOHAN DUVAL

le vendredi 7 avril 2017

Titre :

Création Automatisée de Scénarios de Formation pour l'Enseignement d'Activités Métier dans un Environnement Informatique Modulaire

---

École doctorale et discipline ou spécialité :
ED MITT : Image, Information, Hypermedia

Unité de recherche :
Laboratoire IRIT, équipe VORTEX

Directeur/trice(s) de Thèse :

JEAN-PIERRE JESSEL
JEAN-YVES PLANTEC

Jury :
Mme DOMITILE LOURDEAUX, Maître de conférence (Rapporteur)
M. RONAN QUERREC, Maître de conférence (Rapporteur)
M. JEAN-JACQUES BOURDIN, Professeur des universités (Membre)
Mme VERONIQUE GAILDRAT, Professeur des universités (Membre)
M. FREDERIC MERIENNE, Professeur des universités (Membre)
M. AXEL REYMONET, Ingénieur de recherche (Membre)

Remerciements

Finalement, après trois années consacrées à l'accomplissement de ma thèse et à la rédaction de ce manuscrit, c'est terminé ! Et ce n'est pas sans un petit pincement au cœur que je passe à autre chose. En effet, les gens qui m'ont entouré durant ces trois ans ont largement contribués à faire des derniers instants de ma vie « d'étudiant » l'expérience qui m'a laissé le plus de satisfaction depuis le début de ma (courte) vie. Cette page de remerciements leur est donc entièrement dédiée.

Je souhaite tout d'abord remercier la société ACTIA Automotive, qui a fait naître les prémices de mon sujet de thèse, et sans qui je n'aurais donc pas pu vivre cette expérience ! Mais ACTIA, c'est surtout pour moi la grande majorité des gens que j'ai pu côtoyer pendant ma thèse, et c'est à eux que je souhaite dire merci. En particulier, j'adresse mes plus sincères remerciements à Axel et Jérôme T., qui m'ont non seulement encadré de manière juste (et pointilleuse) tout au long de mon doctorat, mais aussi et surtout avec qui je partage beaucoup d'affinités et avec qui les différentes réalisations de cette thèse (ce manuscrit y compris) ont pu être réfléchies et effectuées dans des conditions amicales sans prise de tête, entrecoupées de (longues) parenthèses sur nos activités (vidéoludiques ou non) du moment ! J'ai adoré travailler avec vous, merci ! Je remercie ensuite Jean-Claude, que je respecte beaucoup pour un tas de raisons, mais principalement pour avoir cru en ces travaux et en moi dès les départ sans jamais fléchir. Merci de m'avoir motivé et boosté quand il le fallait (principalement lors de la rédaction de ce manuscrit), et merci d'être resté toi-même en toute circonstance ; bien que mon expérience professionnelle soit limitée, cela fait de toi le directeur le plus proche humainement de ses équipes et de ses employés que je connaisse. Ne changes rien, et merci encore ! Enfin, un grand merci à (en essayant de n'oublier personne) : Axel (encore), Jérôme T. (encore), Olivier D., Jean-Claude (encore), Régis, Christophe, Christine, Olivier B., Cédric, Gauthier, Vincent T., Alexandre, Vincent V., Jérôme D., Thierry, Rémi, Raphaël, Eelco, Jean-Christophe P., Jean-Edouard, Myriam, Jean-Christophe F., Mathias, Pascal, Vincent P., Philippe et Renaud. Merci à vous tous (et à ceux que j'aurais involontairement pu oublier), pour avoir contribué directement ou indirectement à la réussite de cette thèse, mais surtout pour m'avoir permis de passer trois années à ACTIA tout simplement inoubliables !

Je remercie ensuite l'IRIT, et plus précisément l'équipe VORTEX, pour m'avoir servi comme point d'appui pour mon encadrement académique tout au long de cette thèse. Plus particulièrement, merci à mon directeur de thèse Jean-Pierre, et à mes encadrants Jean-Yves et David, pour leurs conseils avisés lors de mes différentes réalisations, aussi bien au niveau académique lors de la rédaction de nos articles et de ce manuscrit, qu'au niveau industriel lors de la conception et l'implémentation des différents environnements informatiques concernés par mes travaux. Je souhaite aussi remercier les différents partenaires industriels et académiques avec qui j'ai eu l'occasion de travailler pendant ces trois ans. Merci donc à la société Opérantis, qui m'a dans un premier temps accueilli pendant une semaine afin de concevoir et réaliser les tous premiers prototypes de Jeux Sérieux qui m'ont servi de base par la suite. Merci à eux pour le temps et l'énergie consacrés au projet Diag'Adventures et à sa réussite. Merci aussi à Cathy, Pierre et David (encore) de l'équipe du SGRL pour m'avoir apporté différentes réflexions sur la forme de notre environnement de formation, aussi bien au niveau ludique que pédagogique.

Enfin, j'adresse un grand merci à l'ensemble de ma famille, qui m'a permis d'en arriver là où j'en suis aujourd'hui. Merci donc à mes parents, à ma sœur, et bien entendu à ma femme,

Pauline, qui a su supporter mes mauvaises humeurs passagères tout au long de ces trois années, mais surtout qui m'a encouragé, rouspété, et même accordé des moments de répits, toujours aux moments opportuns. Cette thèse et ce manuscrit sont pour toi !

Table des matières

Résumé

Les travaux effectués durant cette thèse s'inscrivent dans une problématique assez commune de nos jours : proposer de nouvelles méthodes de formation afin de s'adapter à l'évolution des besoins de la société et aux nouvelles technologies disponibles. En particulier, notre étude se focalise sur le besoin de formation à des activités métier de divers domaines (médical, automobile, aéronautique, etc.) de plus en plus complexes. Alors qu'il existe à l'heure actuelle des outils adaptés pour l'enseignement d'activités impliquant des actions avec des objets du monde réel (jeux sérieux), et d'autres outils adaptés pour l'enseignement d'activités se déroulant intégralement dans une application informatique (tutoriels de logiciel), il n'existe pas d'outil permettant d'enseigner des activités mêlant ces deux types d'opération. Dans ce mémoire, nous décrivons donc tout d'abord les différents travaux menés afin de répondre à cette première problématique. Après avoir étudié les caractéristiques respectives des Jeux Sérieux et des tutoriels de logiciel, nous formalisons l'outil de formation que nous avons conçu afin d'homogénéiser l'utilisation de ces deux catégories d'application dans un seul et même environnement. Nous introduisons alors les notions d'Environnement de Formation et de Scénario de Formation, qui font respectivement la distinction entre l'environnement dans lequel évoluent les apprenants au fur et à mesure des leçons, et la description formelle des activités à réaliser ainsi que des éléments pédagogiques et ludiques qui leurs sont liés.

Cependant, le développement d'un tel outil représente une tâche particulièrement ardue pour les formateurs qui ne possèdent souvent pas l'expertise informatique requise. Cette observation vient en contradiction avec le fait que ces mêmes formateurs doivent être impliqués dans le processus de développement, du fait qu'ils possèdent à la fois l'expertise métier et l'expertise pédagogique liées aux activités métier à enseigner. Ainsi, l'étude de méthodes et d'environnements permettant de faciliter la création de tels outils constitue un deuxième axe de recherche de notre thèse. Suite à l'analyse un ensemble de langages textuels et graphiques, nous proposons dans un premier temps un langage dédié permettant de décrire nos Scénarios de Formation. Nous introduisons dans un second temps la représentation associée qui, à l'aide de plusieurs entités graphiques aux formes et à l'utilité bien définies, permet aux formateurs de décrire des scénarios en adéquation avec leurs compétences. Enfin, nous présentons l'environnement auteur permettant de faciliter la tâche aux formateurs grâce à diverses fonctionnalités d'automatisation et de capitalisation. L'ensemble de ces concepts forme la méthodologie globale que nous proposons dans cette thèse afin de rendre accessible la création de scénarios de formation aux formateurs.

Finalement, ces différents travaux sont illustrés au travers de plusieurs applications ayant été implémentées afin de mettre en application nos contributions dans le contexte industriel du diagnostic automobile. Nous présentons et évaluons alors l'outil de formation et l'environnement auteur correspondants qui, ensemble, garantissent un processus industriel complet et plausible, depuis la création d'un scénario de formation jusqu'à son exécution.

Mots-clés : Environnement auteur, Formation, Jeux sérieux, Tutoriels de logiciel, Diagnostic automobile

Abstract

The work carried out during this PhD thesis is related to a common issue these days: offering new training methods to adapt to the evolution of our society and to the new available technologies. Our study focuses on the necessity of training professional activities which take place in various domains (health, automotive, aeronautics, etc.), and which are becoming more and more complex. Nowadays, there are tools that are adapted to the training of activities involving interactions with objects from the real word (serious games), and there are other tools that are suitable for training activities which only involve the use of one or several business software applications (software tutorials). However, there are no fitting tools for training activities which involve both types of operation. In this manuscript, we first describe the work we performed to bring an answer to this problematic. After having studied the respective features of serious games and software tutorials, we formalize the training tool that we have designed to homogenize the use of these two application categories in a unique computer environment. We then introduce two notions. The first one is the notion of Training Environment, which stands for the environment in which trainees will progress to accomplish the different lessons being available through the tool. In a complementary manner, we define the notion of Training Scenario, which precisely is the formal description of one lesson in the Training Environment, with all its components: activities to be performed, pedagogical elements, and playful elements.

However, the implementation of such a tool is a very complex task for trainers which often do not have the required computing expertise. This observation comes in contradiction with the fact that these very trainers must be involved in the development process, because they are the ones who own the professional expertise and the pedagogical expertise associated with the activities to be trained. Thus, the study of methods and environments easing the creation of such tools represent a second research axis for this thesis. After having analyzed a set of textual and graphical languages, we first propose a Domain Specific Modeling Language allowing the description of our Training Scenarios. Second, we introduce the associated representation which, thanks to several graphical entities well-defined, allows trainers to describe their own scenarios in line with their skills and expertise. Last, we present the authoring tool that allows to ease the scenario description task for trainers, thanks to various features which aim at automating the process and promoting reutilization. These concepts shape the global methodology that we propose in this manuscript to make training scenarios creation in trainers reach.

Finally, these works are illustrated through the implementation of various applications which aims at putting into practice our contributions in the industrial context of automotive diagnostic. We then present and evaluate the corresponding training tool and authoring tool which together assure a complete and plausible industrial process, from the training scenario description to its execution.

Keywords: Authoring tool, Training, Serious games, Software tutoring, Automotive diagnostic

Introduction

Contexte Scientifique

Depuis toujours, les méthodes de formation n'ont eu de cesse d'évoluer afin de s'adapter aux besoins de la société et aux technologies disponibles. Avec la démocratisation de l'Informatique et de l'Internet, de nouveaux types d'outils ont fait leur apparition, tels que les logiciels éducatifs ou, plus récemment, les solutions e-learning. Les avantages apportés par de tels outils sont multiples. En particulier, ils entrainent une implication et une motivation plus forte chez l'apprenant, de par le fait qu'ils lui permettent de devenir l'acteur principal de sa formation, et qu'ils permettent de proposer une certaine forme d'interactivité et par conséquent d'attractivité. De plus, ils permettent de former un nombre potentiellement illimité d'apprenants, ce qui en fait un outil de choix pour les entreprises devant organiser des sessions de formation qui ciblent souvent une masse importante de salariés. Il n'est ainsi pas étonnant d'observer que l'utilisation de tels outils se répand de plus en plus depuis plusieurs années.

En parallèle, les chercheurs se sont intéressés à l'étude de moyens pour proposer aux apprenants des méthodes de formations toujours plus en adéquation avec leurs attentes, c'est-à-dire des méthodes leur donnant toujours plus l'envie d'apprendre. Une solution retenue et de plus en plus utilisée aujourd'hui concerne l'intégration d'éléments ludiques, extraits de la notion générale de *jeu* et plus spécifiquement des produits vidéoludiques. Les jeux sérieux sont alors certainement l'outil de formation le plus représentatif de cette idée, du fait qu'il soit initialement conçu pour intégrer de manière cohérente les deux dimensions pédagogiques et ludiques. Dans le cadre de nos recherches, nous nous sommes donc tout d'abord intéressés à analyser les différentes caractéristiques de chacune de ces dimensions. Nous avons ensuite étendu notre analyse à l'étude des tutoriels de logiciel, qui représente un second type d'outil de formation ayant été ciblé pour introduire des éléments de jeu. L'originalité de nos travaux réside par la suite dans le fait que nous ayons conçu un outil de formation qui combine de manière homogène les particularités de ces deux catégories d'outil à la fois pédagogique et ludique, afin de répondre à un besoin industriel bien spécifique.

Une seconde piste de recherche en rapport avec l'utilisation de cette nouvelle génération d'outil de formation concerne l'étude des processus et des environnements permettant de faciliter leur développement. En effet, leur nature informatique fait qu'ils sont intrinsèquement plus complexes à créer que des sessions de formations plus traditionnelles basées sur des livres ou des diaporamas. Paradoxalement, il est nécessaire d'impliquer les personnes possédant l'expertise pédagogique et les connaissances d'un domaine particulier dans le processus de développement afin d'obtenir à la fin un outil de formation pertinent. Ces personnes n'étant pas nécessairement des experts en informatique, il y a donc une nécessité de leur proposer les outils adaptés pour qu'ils puissent effectivement participer à leur développement. Dans le cadre de nos recherches, nous effectuons donc un tour d'horizon des outils existant - langages, méthodologies, environnements auteur - chacun ciblant des aspects particuliers du processus de développement global. La finalité de cette étude est de pouvoir proposer notre propre méthodologie qui permettra à des formateurs de décrire formellement les leçons étudiées au travers de notre outil de formation.

Bien que les travaux effectués durant cette thèse aient été fortement guidés par le contexte industriel dans lequel elle s'est déroulée (voir section suivante), le lecteur aura à la fin de ce mémoire les outils nécessaires pour appliquer notre étude au domaine qui l'intéresse. Il sera ainsi en mesure d'implémenter d'un côté l'outil de formation qui lui convient, et de l'autre l'environnement auteur et tous les processus qui lui sont liés pour décrire formellement les leçons associées. Finalement, il pourra apprécier la chaîne industrielle complète mise en évidence, depuis la création de nouvelles leçons jusqu'à leur exécution.

Contexte Industriel

L'ensemble des travaux menés au cours de cette thèse ont été réalisés dans le cadre d'un partenariat entre l'équipe VORTEX (Visual Object : from Reality to Expression) de l'IRIT (Institut de Recherche en Informatique de Toulouse) et la société ACTIA, dont l'activité principale concerne la conception et la fabrication de composants électroniques pour plusieurs domaines. En particulier, la division Automotive d'ACTIA se concentre sur la production de système embarqués et de systèmes de diagnostic pour tout type de véhicule : léger, poids lourd, bus et car, etc… Tel que nous allons le présenter par la suite, nos recherches s'inscrivent dans une problématique générale rencontrée par les différents acteurs de ce domaine : fournir aux garagistes des outils pour les aider à diagnostiquer et à réparer les pannes qu'ils rencontrent.

Depuis plusieurs années, dans une volonté d'amélioration continue de l'expérience de conduite, les constructeurs automobiles proposent des véhicules aux fonctionnalités toujours plus pratiques et sécurisantes pour le consommateur. C'est ainsi que nous avons vu apparaître des systèmes aujourd'hui devenus courants, tels que par exemple le GPS, la caméra de recul, ou encore l'allumage automatique des phares. Ces innovations, rendues possibles grâce à l'ajout de nombreux composants électroniques et mécaniques, ont en contrepartie contribué à complexifier la structure générale des véhicules. Par conséquent, la tâche des garagistes pour 1) diagnostiquer des pannes/disfonctionnements sur ces véhicules, puis pour 2) les réparer, s'est elle aussi complexifiée. Malgré les outils de diagnostic qui accompagnent les garagistes dans leur travail quotidien, cela a conduit à une hausse des coûts associés :

- pour le constructeur (appels à l'assistance plus fréquents de la part des garagistes),
- pour les garages (augmentation du temps passé pour résoudre une panne),
- et finalement pour le client final.

Afin de répondre à cette problématique, le département de recherche de la société ACTIA a déjà mis en place par le passé plusieurs projets dont l'objectif général était de proposer des solutions permettant d'assister les garagistes dans les tâches de diagnostic qu'ils se voyaient confier. Parmi ces projets, nous pouvons citer OBIR (Ontology Based Information Retrieval), qui a donné lieu à la thèse (Reymonet, 2008). Les travaux associés visaient à *"fournir aux garagistes un moyen simple et intuitif* […] *de vérifier dans une base de fiches d'incidents si la panne traitée a déjà été précédemment résolue* […]*"*, pour qu'ils puissent ensuite éventuellement s'appuyer dessus et résoudre le problème rencontré. Cela s'est alors traduit par la mise en place d'un moteur de recherche destiné au garagiste, et d'un environnement auteur permettant d'enrichir la base de fiches d'incidents et le modèle ontologique associé. De manière générale, le projet OBIR et les autres projets menés jusqu'à maintenant ont permis de proposer des solutions visant à assister le garagiste **pendant** sa tâche de diagnostic.

Dans le cadre de cette thèse, nous avons souhaité nous intéresser à des solutions dont l'objectif serait de fournir au garagiste les connaissances nécessaires à l'accomplissement d'une activité de diagnostic **en amont** de l'opération réelle (sur le véhicule client). Malgré les formations reçues annuellement par les garagistes, nous avons pu observer lors d'une étude que certaines pannes et procédures de réparation associées étaient complexes et coûteuses à mettre en place, et n'étaient donc jamais réalisées entièrement. De plus, nous avons pu noter qu'un temps relativement long pouvait s'écouler entre le moment où les garagistes recevaient une formation liée à une nouvelle fonctionnalité, et le moment où ils rencontraient effectivement un cas de panne associé dans leur garage. Dans ce cas, il est alors probable que le garagiste ait oublié la formation reçue et qu'il doive donc faire appel à l'assistance afin de diagnostiquer et de réparer la panne. La Figure 1 illustre ainsi l'écart qui se créé au fil du temps entre l'augmentation de la complexité des véhicules et l'augmentation des compétences des garagistes. Nos travaux visent donc à combler ce besoin de formation qui apparait.

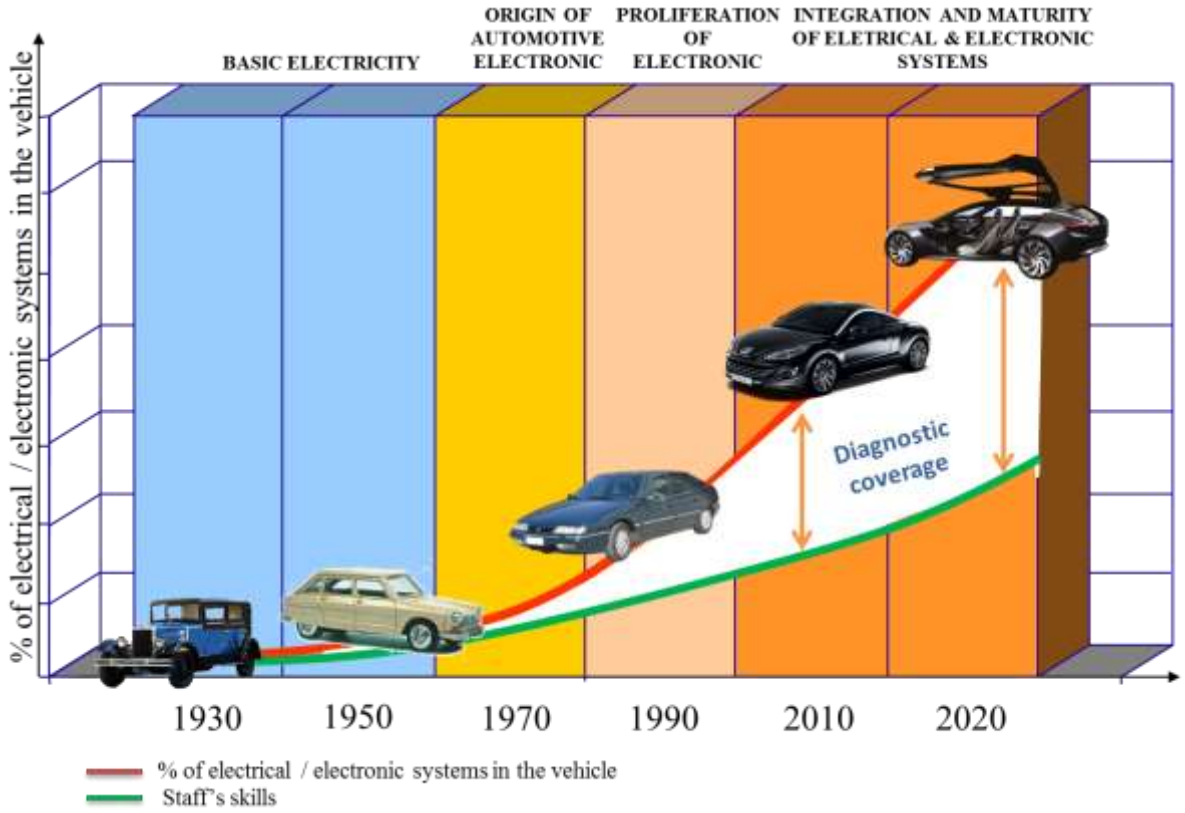


FIGURE 1. ILLUSTRATION DE L'ECART QUI SE CREE ENTRE L'AUGMENTATION DE LA COMPLEXITE DES VEHICULES ET L'AUGMENTATION DES COMPETENCES DES GARAGISTES

Une solution qui intéressa initialement ACTIA fut l'introduction d'outils de formation de type jeux sérieux dans le processus de formation des garagistes. En parallèle, un second souhait était d'intégrer le logiciel de diagnostic, représentant une partie importante des activités de diagnostic étudiées, directement dans l'outil de formation. En effet, cela permettrait d'un côté de pouvoir jouer les différentes leçons quelle que soit la langue utilisée (sans avoir besoin d'effectuer des captures d'écrans du logiciel pour chacune des langues), et de l'autre de pouvoir les maintenir à jour plus simplement au fur et à mesure des évolutions effectuées sur le logiciel de diagnostic. Cela nous a alors mené à établir la première problématique industrielle de nos travaux : ***concevoir***

un outil de formation permettant d'enseigner des activités métier dans un environnement informatique mêlant le logiciel de diagnostic avec les éléments fondamentaux des jeux sérieux.

Finalement, un troisième souhait d'ACTIA était de pouvoir être réactif face aux besoins exprimés par des clients potentiels, et ainsi de leur fournir ce type d'outil de formation le plus rapidement possible à chaque demande. Nous avons alors choisi de concentrer nos travaux sur l'automatisation du processus de description formelle d'une leçon enseignée par notre outil de formation, dans la finalité de pouvoir impliquer les formateurs directement dans le développement de l'outil. De cette volonté en a alors découlé la deuxième problématique étudiée dans ce mémoire : ***définir une méthodologie orientée-formateur pour la description formelle d'une leçon étudiée au travers de notre outil de formation***.

Organisation du Mémoire

Les quatre premiers chapitres de ce mémoire sont organisés afin de répondre de manière générique aux problématiques soulevées précédemment. Ainsi, nous effectuons dans le 0 un tour d'horizon à la fois des jeux sérieux et des tutoriels de logiciel, de manière à mettre en évidence leurs caractéristiques respectives qui devront être prises en compte lors de la conception de notre outil de formation. Puis, nous étudions dans le 0 les langages, représentations graphiques, ou autre méthodologies / environnements permettant d'assister les utilisateurs dans leur tâche de développement de jeux sérieux ou de tutoriels de logiciel. Cela nous permettra alors d'isoler les outils qu'il nous a semblé intéressant d'appliquer dans le cadre de nos travaux.

Le 0 est ensuite pour nous l'occasion de présenter l'outil de formation que nous avons conçu à partir de notre Etat de l'Art afin de répondre à la première problématique. Nous introduisons alors les notions d'Environnement de Formation et de scénario qui représentent les éléments centraux sur lesquels s'appuient l'ensemble de nos travaux. Puis, nous présentons dans le Chapitre 4 la méthodologie que nous proposons afin de rendre la création d'un scénario accessible aux formateurs, avec l'ensemble des fonctionnalités et des processus qui la caractérisent.

Finalement, nous nous attachons à présenter puis à évaluer dans le 0 les différentes applications ayant été implémentées (i.e. outil de formation et environnement auteur) afin de mettre en application nos différentes contributions dans le contexte industriel du diagnostic automobile.

Partie I. État de l'Art

Chapitre 1. Enseignement d'activités métier : vers un outil de formation interactif et ludique

1.1 Motivations

Comme nous l'avons expliqué dans le chapitre d'introduction, les travaux qui ont été menés dans le cadre de nos recherches s'intéressent à l'enseignement d'activités qui nécessitent l'interaction avec le monde réel et avec un ou plusieurs logiciels. Ainsi, nous avons dans premier temps effectué un tour d'horizon des différents outils de formation existants afin de comprendre quelles en sont les caractéristiques et ensuite introduire la solution qui satisfait à nos besoins. Entre tutoriels traditionnels sous forme de documentations (Carroll & Rosson, 1987) (Tuck & Olsen, 1990) ou de vidéos en ligne (Grossman & Fitzmaurice, 2010) (Pongnumkul, et al., 2011), et outils plus évolués tels que les plateformes d'E-Learning (Rosenberg, 2001), les solutions possibles sont multiples. Dans un souci de rendre les sessions de formation plus engageantes pour l'apprenant, nous nous sommes alors intéressés plus particulièrement aux outils de formation interactifs. L'objectif principal est d'impliquer au maximum les apprenants (Li, Grossman, & Fitzmaurice, 2012) en adoptant un processus d'apprentissage actif et ainsi d'améliorer l'efficacité des enseignements (Benware & Deci, 1984). Une deuxième piste toute aussi importante pour nos recherches a consisté à nous intéresser aux outils qui apportent une expérience ludique en parallèle de leur vocation pédagogique initiale. En effet, il est aujourd'hui reconnu que ce caractère ludique, intégré dans des outils de formations, possède des vertus pédagogiques directement liées à la motivation intrinsèque qu'il amène chez l'utilisateur (Li, Grossman, & Fitzmaurice, 2012) (Mouaheb, Fahli, Moussetad, & Eljamali, 2012).

Les premières techniques de formation auxquelles nous nous sommes donc intéressés et qui respectent nos critères sont les jeux sérieux. Nous introduisons puis définissons en détail ces outils dans la partie 1.2. L'utilisation des jeux sérieux dans notre contexte est d'autant plus pertinente qu'ils sont de plus en plus utilisés dans le monde professionnel en raison de leur capacité à recréer fidèlement des situations de formation proches de situations réelles (Mouaheb, Fahli, Moussetad, & Eljamali, 2012). En particulier, ils permettent de reproduire des environnements et des situations complexes pour des coûts humains et matériels moindres. Ainsi, les jeux sérieux ont vu leur utilisation se démocratiser et s'appliquer à de nombreux contextes professionnels (Zyda, 2005) tels que la santé (Thompson, et al., 2008), l'aéronautique (Lépinard, 2014) ou encore l'automobile (Duval, et al., 2015). Cela montre alors que ces outils ont atteint un niveau de maturité suffisant et que nous pouvons donc considérer leur utilisation dans notre contexte industriel.

Actuellement, l'usage des jeux sérieux dans le cadre de la formation à des procédures métier se limite souvent aux procédures qui impliquent d'interagir avec des objets du monde réel. Si la procédure implique aussi d'interagir avec une application informatique, seules les fonctionnalités nécessaires seront reproduites à minima dans l'environnement. C'est pourquoi nous avons souhaité étendre notre étude aux tutoriels de logiciel, qui existent depuis l'apparition des premières Interfaces Hommes-Machines (IHM) (Li, Grossman, & Fitzmaurice, 2012). En dehors des systèmes de tutoriel classiques dont nous ferons un bref résumé en 1.3.2.1, nous nous sommes

intéressés aux systèmes plus interactifs, qui sont la plupart du temps intégrés directement dans le logiciel concerné (1.3.2.2). Ces outils permettent de fournir des formations plus riches à l'apprenant en le guidant de façon contextuelle dans l'utilisation du logiciel au fur et à mesure de la procédure (Grossman & Fitzmaurice, 2010) (Grabler, Agrawala, Li, Dontcheva, & Igarashi, 2009). Une nouvelle méthode visant à augmenter le niveau d'engagement de l'élève lors de l'utilisation de ce type d'outil de formation est la *gamification* (1.3.2.3). Il s'agit d'intégrer des éléments de jeu dans un contexte initialement non-ludique (Deterding, Dixon, Khaled, & Nacke, 2011). Dans le cadre des tutoriels de logiciels, cela permet de renforcer l'interactivité entre l'apprenant et l'outil d'apprentissage, et de captiver son intérêt pour une formation qu'il n'aurait peut-être pas effectué autrement (Li, Grossman, & Fitzmaurice, 2012).

1.2 Des outils pour l'enseignement de procédures dans le monde réel : les jeux sérieux

Dans une optique de former des professionnels à des procédures complexes dans le monde réel, nous avons souhaité étudier les jeux sérieux pour leurs apports en termes d'interactivité et d'apprentissage ludique. Afin d'introduire l'outil de formation final qui nous intéressera dans les chapitres suivants, nous avons tout d'abord souhaité comprendre quelles en étaient les caractéristiques fondamentales. Dans cette partie, nous introduirons donc ce que sont les jeux sérieux à partir de définitions existantes (1.2.1), et donnerons des exemples de domaines dans lesquels ils sont utilisés. Nous étudierons ensuite quels sont les critères à respecter afin d'obtenir des outils de formations efficaces, du point de vue pédagogique (1.2.2) et ludique (1.2.3).

1.2.1 Jeux sérieux : une vue d'ensemble

1.2.1.1 Définition

Il est possible de trouver dans la littérature différentes définitions de ce que sont les jeux sérieux. En particulier, (Alvarez, 2007), dont la thèse entière est dédiée au concept même du Serious Game (jeu sérieux en anglais), les définit comme tels en conclusion de son premier chapitre :

> *Application informatique, dont l'intention initiale est de combiner, avec cohérence, à la fois des aspects sérieux (Serious) tels, de manière non exhaustive et non exclusive, l'enseignement, l'apprentissage, la communication, ou encore l'information, avec des ressorts ludiques issus du jeu vidéo (Game). Une telle association, qui s'opère par l'implémentation d'un "scénario pédagogique", qui sur le plan informatique correspondrait à implémenter un habillage (sonore et graphique), une histoire et des règles idoines, a donc pour but de s'écarter du simple divertissement. Cet écart semble indexé sur la prégnance du "scénario pédagogique".*

Cette définition est ensuite complétée par celle du *scénario pédagogique* :

> *Fonction dédiée à un "objectif pédagogique", dont la propriété est de susciter l'envie d'apprendre et dont la réalisation dépend d'un jeu vidéo avec lequel elle puisse s'intégrer.*

Dans le même esprit, nous donnons la définition issue des travaux du chercheur (Amato, 2007):

Les « serious games » peuvent être définis comme étant des jeux vidéo utilitaires, c'est-à-dire productifs, dont la conception vise à opérer une transformation chez leurs destinataires allant dans le sens d'une amélioration des compétences (entraînement), de l'adaptation au milieu (traitement des phobies), de la compréhension d'un phénomène (éducation) ou d'une plus grande adhésion au message véhiculé (promotion, publicité, jeux vidéo idéologiques, dits aussi political games).

On voit apparaitre dans ces deux définitions deux grands concepts indissociables qui caractérisent les jeux sérieux et que l'on retrouve dans d'autres définitions de synthèse telle que celle de (Djaouti, 2011) ou de (Mouaheb, Fahli, Moussetad, & Eljamali, 2012) :

- La dimension "sérieuse". Elle concerne un vaste nombre d'éléments, mais en nous limitant à notre contexte, elle concerne tout ce qui est en rapport avec l'apport d'informations et de supports pédagogiques pour enseigner à l'apprenant des activités liées à un domaine professionnel particulier.
- La dimension "ludique". Ce sont tous les éléments qui sont empruntés directement du jeu-vidéo et qui vont donc apporter de l'interactivité dans l'outil de formation et améliorer le niveau d'intérêt et d'attention de l'apprenant vis-à-vis de la leçon et du contenu pédagogique.

1.2.1.2 Un outil de formation en plein essor

Apparus en 2002 avec la sortie de *America's Army* aux États-Unis, les jeux sérieux ont depuis vu leur utilisation largement augmentée. Le schéma ci-dessous (extrait de la thèse de (Djaouti, 2011)) montre en effet l'évolution du nombre de jeux sérieux publiés chaque année jusque 2009 :

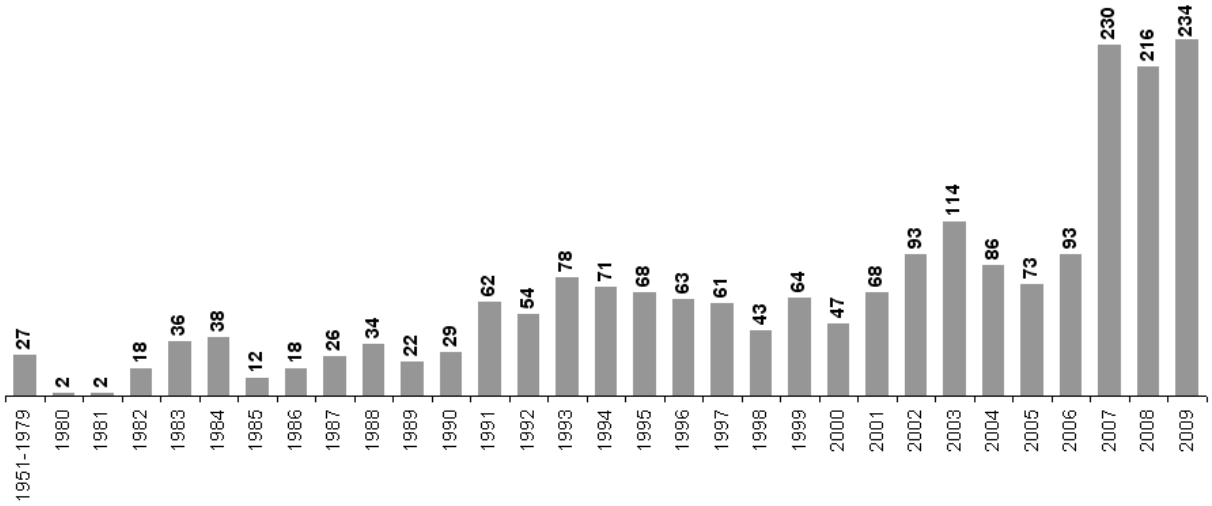


FIGURE 2. ÉVOLUTION DU NOMBRE DE JEUX SERIEUX PUBLIES CHAQUE ANNEE

On observe en effet qu'en 2009, la publication de jeux sérieux a plus que doublé par rapport à 2002. On observe aussi que la création de jeux sérieux est bien antérieure à 2002. Il s'agit en fait d'outils de formation qui utilisaient déjà les codes de conception des jeux sérieux et qui entrent aussi dans la définition donnée dans la partie précédente. Le chiffre d'affaire mondial du marché des jeux sérieux a donc augmenté en conséquence, et continue de croitre depuis 2010 comme

l'atteste la Figure 3 [1]. Cela témoigne donc bien de l'engouement autour de l'utilisation d'applications informatiques ludiques dans le but de faire passer un message sérieux, et ce depuis les débuts de l'informatique et des jeux vidéo.

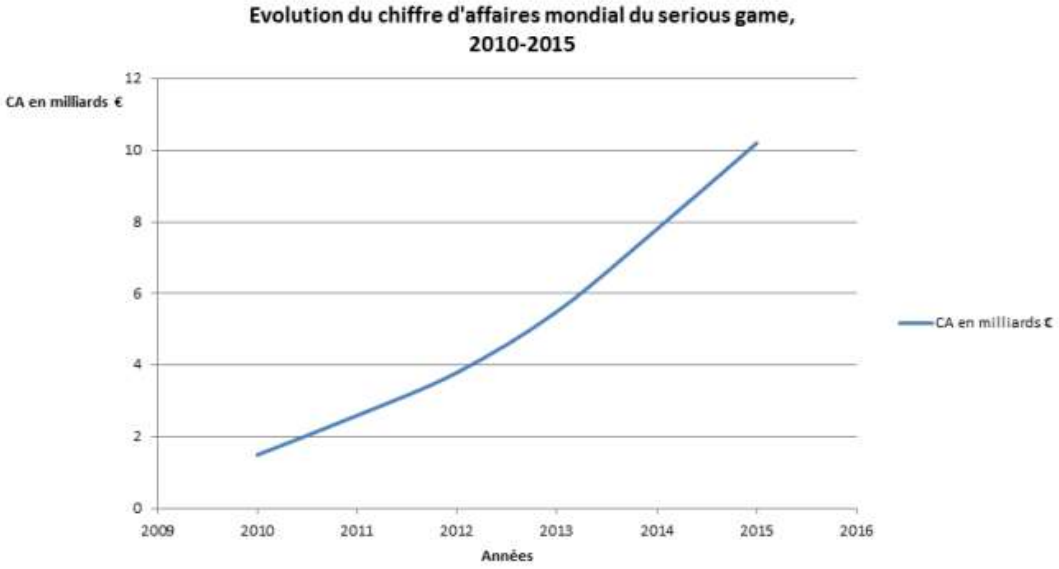


FIGURE 3. EVOLUTION DU CHIFFRE D'AFFAIRES MONDIAL DU SERIOUS GAME ENTRE 2010 ET 2015

De nos jours, les jeux sérieux sont ainsi utilisés dans de nombreux domaines, ce qui est résumé en détail par (Susi, Johannesson, & Backlund, 2007). On peut citer dans un premier temps ceux dédiés au secteur ***militaire***. Dans cette catégorie, le jeu sérieux le plus célèbre est *America's Army*[2]. Conçu et développé par l'armée américaine, son objectif initial était de trouver une nouvelle approche afin de recruter de nouveaux soldats volontaires. Pour les futurs recrutés, *America's Army* (Army, 2003) est aussi un outil permettant de les pré-entrainer au métier de soldat. Pour cela, le jeu reproduit fidèlement un nombre important d'éléments liés au métier de soldat, telles que les armes ou les véhicules. Les mécaniques de jeu sont ensuite très proches de celles d'un jeu de tir classique à la première personne, à la différence près qu'il y a une volonté ici de proposer des sensations et des environnements les plus proches possible de la réalité. C'est en partie pour cela que ce nouvel outil a été un succès. A présent, il est aussi considéré comme un jeu vidéo à part entière avec une très grande communauté, du fait du plaisir qu'il procure à ses utilisateurs.

Une deuxième catégorie de jeux sérieux concerne ceux qui s'intéressent au domaine ***politique***. Ils peuvent comprendre un large panel d'applications, allant de la gestion d'une ville et des différents problèmes qu'elle peut rencontrer (trafic routier, gestion de budget, …), à la gestion de crises nationales voire internationales (sensibilisation à des maladies contagieuses, combattre le terrorisme, …). Dans cette catégorie, on peut citer le jeu sérieux de Gonzalo Frasca, *September 12th*[3], dans lequel il était question de sensibiliser le public aux causes possibles de la montée du terrorisme (suite aux attentats du 11 septembre). La forme de l'application, très simple dans ses mécaniques et utilisant une représentation assez *cartoon*, permet au final de faire réfléchir le

[1] http://gamesforhealth.jimdo.com/digital-sant%C3%A9/le-march%C3%A9-des-serious-games-1/

[2] https://www.americasarmy.com/

[3] http://www.newsgaming.com/games/index12.htm

grand public sur un sujet d'actualité sensible (Djaouti, 2011) en ayant recours à des concepts issus du jeu vidéo.

Viennent ensuite les jeux sérieux à portée ***éducative***. Ce sont ceux qui sont majoritairement utilisés dans un contexte scolaire afin d'enseigner certains principes théoriques et de les de les mettre en pratiques à travers un environnement ludique. Les sujets d'enseignement sont ici aussi divers et variés : programmation (Muratet, Torguet, Jessel, & Viallet, 2009), génétique (Annetta, Minogue, Holmes, & Cheng, 2009), développement logiciel (Hainey, Connolly, Stansfield, & Boyle, 2011) ou encore mathématiques (Kebritchi, Hirumi, & Bai, 2010). La forme des jeux sérieux correspondants est donc différente à chaque fois, mais ont tous un point commun : l'aspect ludique est considéré comme étant tout aussi important que l'aspect pédagogique initial, et la combinaison de ces deux dimensions forment ainsi un environnement d'apprentissage divertissant cohérent. L'utilisation de ce type d'applications permet à chaque fois d'attirer l'attention des étudiants plus efficacement que des supports d'enseignement classiques (manuels, diaporamas, …), et donc d'obtenir une meilleure implication de leur part, et éventuellement de meilleurs résultats quant aux connaissances acquises.

Enfin, une dernière catégorie concerne les jeux sérieux utilisés dans un contexte ***professionnel***, qui visent à former les acteurs d'une industrie particulière à différents types d'activités en rapport avec leur métier (Michael & Chen, 2005) : travail d'équipe, communication, apprentissage de procédures particulières plus ou moins complexes, ou encore mise en application de savoir-faire théoriques. De l'aéronautique à l'automobile en passant par le commerce, les secteurs professionnels sont nombreux. Parmi ceux-ci, celui de la ***santé*** est particulièrement concerné du fait que les opérations à effectuer sont souvent critiques avec la santé du patient en jeu. Durant nos recherches, nous avons été amenés à rentrer en contact avec certains acteurs qui développent des jeux sérieux dans le domaine de la santé, tels que la société MEDUSIMS[4] et le Serious Game Research Lab de l'Université Jean-François Champollion à Albi. Afin de remplir leur objectif pédagogique, les jeux sérieux développés (Panzoli, et al., 2014) proposent un environnement virtuel très réaliste et immersif. La dimension ludique est assurée par une mise en scène dynamique, ainsi que des mécaniques de jeu empruntées des jeux vidéo d'aventures dans lesquels l'exploration et la recherche d'indice sont des éléments clés dans la réussite de la mission. Cette catégorie de jeux sérieux nous intéresse tout particulièrement puisqu'elle correspond directement au contexte de notre étude. Nous nous appuierons donc sur certaines de leurs caractéristiques pour définir notre outil de formation.

Bien qu'il y ait un manque de preuves concrètes permettant d'affirmer avec certitude que les jeux sérieux sont des outils de formation plus efficaces que des méthodes plus traditionnelles (Girard, Ecalle, & Magnan, 2013), on observe malgré tout qu'ils permettent d'augmenter sensiblement le degré d'engagement et d'implication des personnes ciblées (étudiants, professionnels, grand public…) (Annetta, Minogue, Holmes, & Cheng, 2009) (Girard, Ecalle, & Magnan, 2013) (Wrzesien & Raya, 2010). De plus, dans la plupart de la littérature, les retours des apprenants sont dans l'ensemble positifs. Dès lors, il nous apparait pertinent d'utiliser ce genre d'outil dans notre contexte industriel. Les parties 1.2.2 et 1.2.3 vont nous permettre d'analyser plus en détail les différentes caractéristiques des jeux sérieux, afin de mieux comprendre quels seront les enjeux à prendre en compte lors de leur conception et création.

[4] http://medusims.com/

1.2.2 La dimension sérieuse

1.2.2.1 La notion d'objectif pédagogique

Au travers des jeux sérieux, les enseignants ou autres instructeurs professionnels ont pour principale intention d'améliorer les formations en transmettant un ensemble d'objectifs pédagogiques (ou objectifs d'apprentissage) (Alvarez, 2007) (Garris, Ahlers, & Driskell, 2002). Un certain nombre de chercheurs ont tenté par le passé de définir la notion d'objectif pédagogique en les catégorisant. En particulier, (Garris, Ahlers, & Driskell, 2002) synthétisent dans leur article le travail de (Gagne, 1984), d' (Anderson, 1982), et de (Kraiger, Ford, & Salas, 1993) pour parvenir à la classification suivante :

- Objectifs pédagogiques de ***savoir-faire***. Ils correspondent à l'enseignement de procédures techniques et/ou de compétences motrices. Plus généralement, ce sont les objectifs liés à l'apprentissage d'actions manuelles de l'apprenant. Dans notre contexte de diagnostic automobile, savoir brancher l'interface de communication au véhicule et à l'ordinateur est un exemple d'objectif pédagogique de *savoir-faire*.
- Objectifs pédagogiques ***cognitifs***. Ils concernent l'enseignement de différents types de connaissances :
  - Les ***connaissances déclaratives***. Ce sont les connaissances factuelles, celles qui permettent de reconnaitre un certain type de situation et de savoir répondre aux problématiques qu'elle pose. Par exemple, savoir reconnaitre qu'une panne de feux de croisement provient de la partie commande (comodo), de l'actionneur (calculateur) ou de la partie actionnée (ampoule), et savoir comment y remédier, est une connaissance *déclarative*.
  - Les ***connaissances procédurales***. Ce sont les connaissances qui permettent de savoir comment accomplir une tâche donnée. Elles diffèrent des objectifs pédagogiques de *savoir-faire* dans le sens où elles n'impliquent pas de savoir accomplir la tâche en question, mais seulement de savoir comment l'accomplir. Un exemple de connaissance *procédurale* est de savoir comment remplacer les ampoules des feux de croisement.
  - Les ***connaissances stratégiques***. Ce dernier type de connaissance est en rapport avec la faculté de savoir appliquer des principes précédemment appris à des situations plus générales ou à des situations nouvelles. Pouvoir analyser de manière critique une situation donnée (ex : panne des clignotants) en se référant à des connaissances liées à d'autres situations semblables (ex : panne des feux de croisement) est un exemple de connaissance *stratégique* (Wood & Stewart, 1987).
- Objectifs pédagogiques ***affectifs***. Il s'agit ici d'enseigner un concept qui est susceptible de changer la manière d'être ou la manière de penser d'un individu. Ils peuvent donc influer sur le sentiment de confiance en soi, l'attitude adoptée vis-à-vis d'une situation donnée, ou encore sur les préférences personnelles sur un sujet quelconque. Un exemple de jeu possédant un objectif pédagogique *affectif* est celui de (Thomas, Cahill, & Santilli, 1997) dont le but est d'améliorer la confiance des étudiants lorsqu'ils proposent à un partenaire d'avoir des relations sexuelles protégées. Cependant, le contexte du diagnostic automobile ne nous semble pas se prêter à ce type d'objectif pédagogique.

(Wouters, van der Spek, & van Oostendorp, 2009) complètent cette liste avec une quatrième catégorie : les objectifs pédagogiques de ***communication***. Comme son nom l'indique, l'enseignement porte ici sur les compétences de communication avec d'autres personnes. Cela peut donc concerner la faculté de coopérer avec ses membres d'équipes lors d'un travail en groupe par exemple, la manière de négocier dans différents contextes (commercial, concession, etc.), ou encore la transmission d'informations et/ou de connaissances à des personnes tierces (ex : compte rendu oral de diagnostic au client).

De son côté, (Alvarez, 2007) donne dans son mémoire de thèse, après avoir répertorié un grand nombre de jeux sérieux et observé les concepts qu'ils enseignent, une définition plus générale de l'objectif pédagogique :

> *Dans le cadre du serious game, l'objectif pédagogique semble s'apparenter soit à l'apprentissage d'une connaissance, soit à celle d'une pratique, ou bien des deux à la fois.*

Bien qu'étant moins précise, elle rejoint la catégorisation de (Garris, Ahlers, & Driskell, 2002) donnée plus haut. Dans le cadre de notre étude, nous retiendrons que cette notion d'objectif pédagogique fait partie intégrante des jeux sérieux et qu'elle permet catégoriser le contenu pédagogique qu'ils mettent en valeur. Il sera donc important pour nous de la laisser transparaitre dans la forme finale de l'outil de formation que nous utiliserons

1.2.2.2 La forme de la pédagogie

A présent, nous savons que la notion d'objectif pédagogique est une notion importante au sein de la dimension "sérieuse" des jeux sérieux. La question qui se pose maintenant est la suivante : "Quel forme de pédagogie, ou encore quels moyens pédagogiques, doit être mise en œuvre afin de transmettre efficacement aux apprenants la connaissance liée aux objectifs pédagogiques du Jeu Sérieux ?". (Alvarez, 2007) nous donne une partie de la réponse dans son mémoire :

> *[...] il semblerait que l'apprentissage actif, et notamment l'une de ses sous catégories, l'Approche Par Projet (APP), soit sur un plan pédagogique compatible avec le serious game.*

Dans la ***pédagogie active***, l'objectif est d'encourager l'apprenant à apprendre par lui-même, à aller chercher l'information dont il a besoin par ses propres moyens (Benware & Deci, 1984) (Alvarez, 2007). Cette forme de pédagogie semble donc en effet adaptée aux jeux sérieux. Plus généralement, elle parait particulièrement adaptée aux outils de formation que nous souhaitons utiliser dans le cadre de nos recherches, puisqu'elle qu'elle suscite un niveau d'engagement important de l'apprenant et une certaine forme d'interactivité. Malgré tout, cette réponse nous semble assez vaste car elle n'insiste pas sur des propriétés précises qui pourraient nous permettre d'appliquer cette approche à n'importe quels jeux sérieux.

En parallèle, certains chercheurs ont souhaité proposer un certain nombre de critères nous permettant alors de définir plus spécifiquement la forme de pédagogie à adopter pour obtenir des jeux sérieux efficaces. Ainsi, (Harteveld, Guimarães, Mayer, & Bidarra, 2007) définissent l'ensemble de propriétés suivant :

- ***Réflexion***. C'est le principe selon lequel l'application doit stimuler une *réflexion* chez l'utilisateur sur les principes théoriques généraux qui peuvent être dégagés à partir des concepts concrets mis en évidence tout au long de la leçon.

- ***Expérience***. Cette notion représente le concept "d'apprendre en faisant". C'est la faculté qu'a le Jeu Sérieux de laisser le joueur apprendre la leçon en réalisant lui-même les actions qui lui sont liées, tout en minimisant la quantité d'informations fournie.
- ***Peu d'éléments superflus***. L'objectif final du Jeu Sérieux étant de transmettre un ensemble d'objectifs pédagogiques (comme admis en 1.2.2.1), il ne faut pas distraire l'attention de l'apprenant vis-à-vis du contenu pédagogique. Pour cela, il est déconseillé d'intégrer des éléments tels que des contraintes de temps non-justifié ou du contenu informatif qui ne serait pas en rapport direct avec la leçon. En effet, de tels éléments peuvent avoir des effets néfastes sur la concentration de l'élève, et donc sur son apprentissage.
- ***Exploration***. Afin de permettre au joueur d'apprendre plus efficacement, il est recommandé de le laisser se familiariser avec l'environnement du Jeu Sérieux. En le laissant explorer librement et interagir avec les différents objets qui composent l'environnement, il sera à même de mieux assimiler les objectifs pédagogiques de la leçon.
- ***Incrémental***. Enfin, la difficulté des principes théoriques à assimiler ou des actions à effectuer doit être progressive afin de ne pas perdre le joueur et d'aller à l'encontre de l'objectif initial de l'outil de formation.

Nous venons donc ici de dresser une liste de recommandations qu'il est souhaitable de suivre afin de concevoir des jeux sérieux efficaces au niveau de la pédagogie. Néanmoins, il nous manque toujours un minimum de pistes quant aux éléments pédagogiques concrets à intégrer dans l'outil. En nous basant sur l'article de (Garris, Ahlers, & Driskell, 2002), nous présentons ci-dessous deux de ces éléments qu'il nous semble importants de respecter :

- ***Feedbacks***. Dans tout environnement de formation, qu'il soit humain ou matériel (logiciel), la notion de feedback est absolument nécessaire. Ces feedbacks peuvent prendre plusieurs formes, tel que l'apport de connaissances ou la présentation de résultats suite à une action (bonne ou mauvaise) de l'élève. Dans tous les cas, ils permettent dans un premier temps d'accompagner l'apprenant tout au long de la leçon, pour éventuellement améliorer ses performances à terme (Kluger, 1996). Dans un second temps, ils permettent d'augmenter le niveau de motivation de l'apprenant, en évaluant continuellement ses progrès vis-à-vis des objectifs pédagogiques de la leçon. Cela le pousse alors à persévérer, en quête d'une compréhension parfaite de la leçon. Intégrer un système de feedback au sein des jeux sérieux nous semble alors indispensable.
- ***Résumé***. Quel que soit le processus de formation envisagé, une phase de résumé est très souvent requise. En effet, c'est lors de cette phase que sera fourni à l'apprenant des explications ou des éclaircissements sur différents points-clé de la leçon. Elle permettra aussi de lui fournir une évaluation de ses performances (Moreno-Ger, Burgos, Martínez-Ortiz, Sierra, & Fernández-Manjón, 2008) afin qu'il se rende compte des points éventuels à améliorer. Lors de cette phase critique, il s'agira alors d'analyser les différents évènements qui se sont produits durant la leçon pour en tirer des conclusions pédagogiques. Dans cette optique, il sera particulièrement intéressant d'insister sur les erreurs de l'apprenant en lui fournissant des explications, tout en lui indiquant comment les éviter. De plus, dans le cadre des jeux éducatifs, cette phase est d'autant plus importante qu'il est nécessaire de faire un lien entre l'environnement virtuel et le monde

réel (en termes d'interactions, mais aussi vis-à-vis des éléments purement ludiques tels que le score ou encore des contraintes temporelles).

Finalement, nous avons pu observer dans cette section 3 niveaux de description de la forme de pédagogie qu'il est intéressant d'adopter dans le cadre des jeux sérieux. Le premier niveau, assez large, consiste à s'appuyer sur le principe de pédagogie active afin d'augmenter le niveau d'implication de l'apprenant. Le deuxième établit une liste de recommandations, plus précises que la simple définition de pédagogie active, qui permettent de concevoir des jeux sérieux efficaces. Enfin, le troisième nous donne quelques pistes sur des éléments concrets à intégrer dans l'outil de formation.

Avec la notion d'objectif pédagogique, nous avons maintenant à notre disposition une base solide en ce qui concerne les caractéristiques importantes de la dimension sérieuse des jeux sérieux. Nous nous reposerons donc en partie sur ces différents éléments lorsque nous définirons la forme de notre outil de formation dans la synthèse de ce chapitre

1.2.3 La dimension ludique

Dans cette partie, nous allons maintenant nous intéresser aux caractéristiques de la dimension ludique des jeux sérieux. De nombreux chercheurs ont établi dans leurs travaux respectifs une liste d'attributs de jeu qui participent à la conception d'un bon Game Design pour les jeux éducatifs. Parmi ces travaux, on peut citer celui relativement récent de (Thillainathan, 2013) dans lequel il récapitule de manière exhaustive les différentes règles de conception des jeux sérieux pour obtenir des applications agréables à utiliser tout en restant engageantes du point de vue pédagogique.

Dans les sous-sections qui suivent, nous allons donc essayer de détailler chaque point qui nous a semblé important dans notre recherche de conception d'un outil de formation interactif adapté à notre contexte industriel. En particulier, nous nous intéresserons aux caractéristiques de l'environnement virtuel, environnement dans lequel évolue l'apprenant tout au long de leçon. Puis, nous mettrons en évidence les éléments directement empruntés du jeu vidéo qui viennent compléter l'environnement virtuel afin de former un ensemble ludique cohérent.

1.2.3.1 Un environnement virtuel immersif et engageant

A la manière des jeux vidéo d'aujourd'hui, les jeux sérieux offrent aux utilisateurs l'opportunité d'évoluer dans un environnement virtuel dans le but d'effectuer un certain nombre d'opérations. Cet environnement virtuel peut prendre plusieurs formes, et donc proposer au joueur autant d'expériences ludiques différentes (nous y reviendrons un peu plus loin). Malgré tout, quelle que soit la forme employée ou le cadre dans lequel il est utilisé (sérieux ou purement ludique), c'est le caractère ***immersif*** et ***engageant*** de l'environnement virtuel qui va permettre d'en mesurer la pertinence ainsi que la cohérence ludique. En effet, une personne immergée dans un univers fictif sera intrinsèquement plus motivée qu'une personne qui ne l'est pas pour réellement se plonger dans le jeu et accomplir de manière efficace les différents objectifs proposés (Annetta, 2010). Ainsi, nous avons dirigé notre étude vers la recherche d'éléments et/ou de critères pouvant caractériser ces notions d'immersion et d'engagement au sein d'un environnement virtuel.

Dans un premier temps, plusieurs chercheurs ont mis en évidence le principe d'***identification***. Il s'agit ici de représenter au joueur, de la manière la plus pertinente et fidèle possible,

l'environnement dans lequel il évolue et effectue des actions (Wilson, et al., 2009). Cette notion de ***représentation*** fidèle concerne d'un côté les composants *physiques* du monde réel, tels que les personnages, les véhicules, ou encore les objets communs nécessaires pour la réalisation des objectifs (Tang, et al., 2008). D'un autre côté, cette notion de *représentation* concerne aussi les composants *abstraits* ou *mentaux*, tels que les sons ou les émotions/sentiments (au sens large) (Prince & Jentsch, 2001). Ce faisant, l'objectif est de permettre au joueur de s'identifier au maximum à ce monde virtuel, et de lui faire ressentir qu'il est un des acteurs principaux participant à l'évolution de l'environnement. Ce concept est un des concepts clé durant la conception d'un jeu vidéo, car il permet au joueur de se considérer comme faisant partie intégrante de cet univers virtuel. C'est cela-même qui garantit l'immersion du joueur dans l'environnement virtuel pour ensuite l'encourager dans sa progression vis-à-vis de ses objectifs et des éventuelles difficultés rencontrées (Annetta, 2010). Certains chercheurs ont d'ailleurs mis l'accent sur ce concept afin de garantir l'implémentation d'un environnement virtuel simulant aussi précisément que possible l'environnement de travail de l'apprenant. C'est le cas par exemple des travaux de (Querrec, 2003), dans lesquels il présente un modèle spécifique (MASCARET) dont l'objectif est de pouvoir simuler fidèlement les phénomènes physiques et les interactions sociales dans un environnement virtuel composé de plusieurs agents.

Un peu en opposition avec le principe d'identification vient ensuite s'ajouter la notion de ***fantaisie*** (Garris, Ahlers, & Driskell, 2002) (Habgood, Ainsworth, & Benford, 2005). Définie comme étant un environnement qui évoque *des images mentales de situations physiques ou sociales qui n'existent pas* (Malone & Lepper, 1987), l'utilisation de la fantaisie dans un jeu vidéo permet de captiver l'attention et la curiosité du joueur afin qu'il s'intéresse de lui-même à ce monde virtuel. Dans un contexte plus sérieux de formation, l'utilisation de la fantaisie dans l'environnement virtuel permet alors de capturer son intérêt et d'améliorer l'efficacité des enseignements de la leçon (Cordova & Lepper, 1996) (Parker & Lepper, 1992). Dans cette même optique d'attiser la curiosité chez l'apprenant, la notion de mystère vient s'ajouter à celle de fantaisie (Garris, Ahlers, & Driskell, 2002). Il s'agit ici d'encourager le joueur à rechercher d'éventuels indices afin de résoudre la ou les problématiques posées par la leçon. Là encore, cela permet d'accentuer le caractère immersif et engageant de l'environnement virtuel, et participe dans le même temps à rendre l'apprentissage *actif*, tel que décrit dans la partie 1.2.2.2.

Un troisième élément qui va venir renforcer l'aspect immersif de l'environnement virtuel est le ***degré de liberté*** donné au joueur pendant la session de jeu / formation. L'apprenant doit pouvoir être libre d'explorer le monde comme il le souhaite, d'agir comme il l'entend par simple curiosité (Sørensen & Meyer, 2007). Il doit être libre d'expérimenter, de réaliser les actions qui lui passent par la tête pour ensuite observer les conséquences qui en résultent, positives comme négatives. Il est aussi important qu'il puisse avoir une forme de contrôle sur le déroulement de la leçon et l'accomplissement des objectifs (Garris, Ahlers, & Driskell, 2002). S'il y a plusieurs manières de parvenir à un même résultat, le joueur doit pouvoir choisir celle qui lui convient le mieux. Au niveau ludique, cette notion de liberté est très importante car en plus d'immerger l'utilisateur dans cet environnement virtuel, elle contribue à lui faire sentir que le monde évolue selon ses actions et qu'il en est donc l'acteur central. Au niveau pédagogique, c'est un point non-négligeable car cela augmente inévitablement son degré d'implication pendant la leçon, et donc l'efficacité des apprentissages.

Enfin, le degré d'immersion dans l'environnement virtuel va fortement dépendre des ***mécaniques de jeu*** proposées (Younis & Loh, 2010) (Yusoff, Crowder, & Gilbert, 2010). Les mécaniques de

jeu correspondent à la manière dont le monde se comporte en fonction d'une situation donnée et/ou des éventuelles ***interactions*** du joueur (Annetta, 2010). La forme de ces interactions va elle-même dépendre du ***genre de jeu*** implémenté par le jeu vidéo. Dans le cadre particulier des jeux sérieux, le genre de jeu choisi sera grandement influencé par le type des objectifs pédagogiques que l'on souhaite mettre en évidence à travers leur utilisation (Yusoff, Crowder, & Gilbert, 2010). Ainsi, de manière non-exhaustive, les jeux sérieux peuvent emprunter les codes de conception d'un jeu de stratégie (Muratet, Torguet, Jessel, & Viallet, 2009), d'action (Army, 2003) ou d'aventure (Torrente, Del Blanco, Marchiori, Moreno-Ger, & Fernández-Manjón, 2010). Proposer des mécaniques de jeux adaptées et intéressantes est donc un élément important à prendre en considération lors de la conception de jeux sérieux. En effet, ce point concerne directement les interactions entre le joueur et l'environnement virtuel. De fait, plus ces interactions seront pertinentes, plus le niveau d'intérêt et d'engagement de l'utilisateur sera élevé.

1.2.3.2 Les concepts vidéoludiques

En parallèle de l'environnement virtuel et de tous les éléments qui participent à le rendre aussi immersif et engageant que possible, il est recommandé de renforcer l'aspect ludique en intégrant divers concepts, aussi directement issus du monde des jeux vidéo. A la différence des caractéristiques intrinsèques de l'environnement virtuel décrites dans la partie précédente, nous étudierons ici des éléments plus concrets qu'il convient d'y ajouter.

En premier lieu, la ***contextualisation*** des jeux sérieux et des objectifs pédagogiques associés est un élément à prendre en compte lorsqu'il s'agit d'accompagner le joueur de manière ludique tout au long de la leçon. Cette contextualisation peut se faire dans un premier temps en introduisant un ***scénario***, c'est-à-dire une histoire de fond ou une ligne directrice, adapté à la leçon considérée (Younis & Loh, 2010). Ainsi, au fur et à mesure de la leçon, le joueur fait progresser l'histoire en même temps qu'il interagit dans l'environnement virtuel, pour finalement parvenir à une conclusion. Cette notion de scénario est présente dans une grande majorité des jeux vidéo, et participe grandement à augmenter l'intérêt porté par le joueur. Afin d'intégrer ce scénario, une méthode simple mais peu ludique est d'orienter et de diriger l'utilisateur en utilisant différents types de message pour introduire l'objectif général de la leçon ou encore les grandes étapes associées. Une autre méthode, plus dynamique et couramment utilisée dans le domaine des jeux vidéo, est l'utilisation de phases scriptées pendant le déroulement du jeu sérieux (Van Eck, 2007). Durant ces phases scriptées, on pourra par exemple mettre en scène des personnages qui entretiennent une conversation afin d'introduire ou de faire progresser le scénario, ou encore animer l'environnement virtuel afin de mettre en évidence des éléments importants à un instant particulier.

Toujours dans l'esprit de contextualiser les jeux sérieux et de scripter la leçon, l'introduction d'***objectif*** à accomplir (Garris, Ahlers, & Driskell, 2002) (Wilson, et al., 2009) (Younis & Loh, 2010) est un élément de *gameplay* largement utilisé dans le monde vidéoludique. En effet, cette notion d'objectif est présente dans une majorité de jeux vidéo, quel que soit leur genre. En particulier, on la retrouve dans les jeux de rôles sous forme de quête (ex : The Witcher 3 : Wild Hunt[5], The Elder Scrolls V : Skyrim[6]), ou encore dans les jeux d'action et d'aventure sous forme

[5] http://thewitcher.com/en/witcher3

[6] http://www.elderscrolls.com/skyrim/

de mission (ex : Call of Duty[7], Runaway[8]). Cela permet de donner au joueur un énoncé général du ou des problèmes qu'il va lui falloir résoudre. Accomplir un objectif se fait alors en réalisant une suite d'étapes, appelées ***activités d'apprentissage*** dans le cadre des jeux sérieux (Yusoff, Crowder, & Gilbert, 2010). La présentation des objectifs et des activités sous-jacentes peut se faire sous plusieurs formes, comme par exemple une vidéo ou simplement un énoncé textuel (Van Eck, 2007). En reprenant l'analogie avec les jeux de rôle, ceux-ci intègrent souvent un journal de quêtes permettant de récapituler la liste des quêtes pouvant être réalisées à un instant précis, ainsi qu'une description plus ou moins détaillées des différentes étapes permettant de mener à bien la *quête* associée. En plus d'être un élément bien ancré dans la culture vidéoludique, la formalisation de la leçon en termes d'objectif et d'activités permet aux apprenants de situer leur progression à tout instant.

Certains chercheurs soulignent aussi l'importance de ***stimuler les sens*** du joueur dans les applications vidéoludiques (Garris, Ahlers, & Driskell, 2002) (Van Eck, 2007). Cela permet en effet d'attirer son attention sur différents éléments tout au long de la session, et donc de maintenir son niveau d'intérêt constant du début à la fin. Cette notion est ainsi d'autant plus importante dans la conception des jeux sérieux, du fait que l'intérêt porté par le joueur à la leçon va fortement influer sur son engagement et donc l'efficacité des enseignements. Cette sollicitation des sens de l'utilisateur se fait principalement en utilisant des stimuli visuels ou auditifs. La notion de contextualisation évoquée dans les paragraphes précédents, avec la mise en scène de personnages ou de plusieurs objectifs bien définis, est un exemple d'éléments stimulants que nous recherchons. De manière générale, l'utilisation d'effets sonores (musiques, sons ponctuels), d'effets graphiques (animations, films) ou encore d'éléments qui suscitent une réflexion chez le joueur (énigmes, obstacles) permet de stimuler efficacement l'attention du joueur et donc son engagement pendant la leçon (Van Eck, 2007).

Un autre élément important durant la conception des jeux sérieux et emprunté du monde vidéoludique est la notion de ***score*** ou de ***succès*** (achievement en anglais) (Sørensen & Meyer, 2007) (Yusoff, Crowder, & Gilbert, 2010). Au même titre que les éléments de feedback et de résumé de la dimension pédagogique (voir 1.2.2.2), ils permettent d'évaluer les performances de l'utilisateur ainsi que les connaissances acquises au cours de la leçon (Wilson, et al., 2009). Dans la plupart des cas, ils sont donc étroitement liés. Le but recherché ici, en plus des retours pédagogiques, est de susciter une forme de plaisir chez l'apprenant, celui de la récompense. En particulier, permettre aux différents apprenants de comparer leurs résultats (Wilson, et al., 2009) est un concept intéressant car cela les incitera à recommencer la session pour essayer d'obtenir un meilleur score. Dans cette optique, l'utilisation d'éléments quantifiables, tels que le temps passé pour terminer la leçon ou encore le nombre d'erreurs effectuées, est une bonne solution.

Enfin, il est important que les jeux sérieux proposent un certain challenge aux utilisateurs (Garris, Ahlers, & Driskell, 2002). En effet, une leçon trop simple ne sera pas particulièrement intéressante pour eux, et cela pourrait donc être contre-pédagogique en les désengageant de la leçon. A l'inverse une difficulté trop élevée pourrait les frustrer et les décourager, et donc avoir le même effet qu'une trop faible difficulté. La complexité des tâches demandées doit ainsi être progressive (Annetta, 2010). Une grande majorité des jeux vidéo proposent plusieurs niveaux successifs avec une difficulté grandissante. De la même manière, un Jeu Sérieux pourra proposer

[7] https://www.callofduty.com/fr/

[8] http://www.runaway-lejeu.com/fr/

différents scénarios, c'est-à-dire des leçons avec des objectifs pédagogiques différents. Les scénarios se débloqueront au fur et à mesure que l'utilisateur terminera avec succès les scénarios précédents, et la complexité des missions proposées sera progressive. Cela permet alors d'apporter les connaissances petit à petit sans ennuyer ni décourager l'apprenant, et donc de maintenir son intérêt pour la leçon ainsi que son niveau d'engagement constants.

1.3 Les tutoriels de logiciel

Après avoir étudié en détail les jeux sérieux et leurs caractéristiques, nous nous apercevons que ce sont des outils parfaitement adaptés pour simuler, dans un cadre d'enseignement de procédures métiers, l'environnement de travail réel de l'apprenant et tous les objets qui le composent. Dès lors, si la procédure en question implique en tout ou partie de manipuler un logiciel, l'environnement virtuel devra reproduire ce logiciel ainsi que les différentes fonctionnalités qu'il propose. Cependant, pour apprendre à utiliser un logiciel, il nous semble bien plus pertinent et efficace de le manipuler directement, en suivant différentes formes d'aide par exemple. Dans cette optique, nous nous sommes intéressés aux différents types de tutoriaux de logiciel qui existent actuellement. De la même manière que pour les jeux sérieux, nous tenterons d'analyser dans cette partie la forme que peuvent prendre les tutoriaux de logiciel afin d'en extraire les concepts importants pour comprendre la forme finale de notre outil de formation.

1.3.1 Du besoin de formation à l'utilisation de logiciels : la notion d'Apprenabilité

Depuis l'apparition des premiers logiciels et des premières Interfaces Homme-Machine, les chercheurs se sont intéressés à divers moyens permettant de former les personnes amenées à s'en servir (Li, Grossman, & Fitzmaurice, GamiCAD: a gamified tutorial system for first time autocad users, 2012). Les premiers logiciels concernés furent les outils de traitement de texte, comme en attestent les travaux de (Carroll & Rosson, 1987). C'est toujours le cas de nos jours, avec par exemple les tentatives de Microsoft pour créer des tutoriaux interactifs[9] accompagnant la suite Office (et notamment Word Office). Mais de nouvelles catégories de logiciels complexes d'utilisation sont aussi venues s'ajouter à la liste des cibles de ce domaine particulier de la recherche. Ainsi, des chercheurs ont proposé des outils de tutoriels de différentes formes pour des logiciels de retouche photo (Photoshop (Chi, et al., 2012), Gimp (Grabler, Agrawala, Li, Dontcheva, & Igarashi, 2009), …) ou encore des logiciels de Dessin Assisté par Ordinateur (AutoCAD (Li, Grossman, & Fitzmaurice, 2012)).

Afin de mesurer la complexité d'utilisation d'un logiciel et ainsi évaluer le degré d'assistance requise pour pallier à cette complexité, les chercheurs se sont penchés sur la notion d'*apprenabilité* (*learnability* en anglais). L'étude menée par (Grossman, Fitzmaurice, & Attar, 2009) résume de manière complète les différentes connaissances du monde scientifique liées à ce concept. Il est difficile de donner une définition exacte de la notion d'*apprenabilité* depuis l'ensemble des définitions présentes dans la littérature. Cependant, la taxonomie présentée dans leurs travaux permet de distinguer l'*apprenabilité initiale* (*inital learnability)* de l'*apprenabilité prolongée* (*extended learnabiliy*). Dans le premier cas, il s'agit de considérer les performances initiales de l'utilisateur avec le logiciel, alors que dans le deuxième cas il est question de

[9] https://www.microsoft.com/en-us/download/details.aspx?id=26531

considérer les changements dans les performances de l'utilisateur avec le logiciel au cours du temps. De plus, l'*apprenabilité* d'un logiciel doit tenir compte du niveau d'expérience général de l'utilisateur avec les ordinateurs, les interfaces homme-machine, ou encore avec des logiciels similaires. Les connaissances de l'utilisateur vis-à-vis du domaine-métier traité par le logiciel forment aussi une dimension à considérer lorsque la notion d'*apprenabilité* est abordée.

Ensuite, afin d'être en mesure d'analyser de manière quantitative et qualitative le caractère d'*apprenabilité* d'un logiciel, (Grossman, Fitzmaurice, & Attar, 2009)catégorisent et listent différentes métriques à partir de travaux précédents :

- Métriques reposant sur les performances liées à l'accomplissement de tâches
    - Pourcentage d'utilisateurs ayant accompli une tâche de manière optimale.
    - Pourcentage d'utilisateurs ayant accompli une tâche sans aucune aide.
    - Capacité de l'utilisateur à accomplir une tâche de manière optimale après un laps de temps donné.
    - Baisse du nombre d'erreurs effectuées sur un intervalle de temps donné.
    - Temps après lequel l'utilisateur accompli avec succès une tâche donnée.
    - Temps après lequel l'utilisateur accompli un ensemble de tâches dans un laps de temps donné.
    - Qualité du travail effectué pendant l'accomplissement d'une tâche, telle une note attribuée par un jury.
- Métriques reposant sur l'utilisation de commandes (opérations unitaires, instructions)
    - Taux de réussite dans les commandes utilisées, après avoir été formé.
    - Hausse du nombre de commandes utilisées sur un intervalle de temps donné.
    - Hausse dans la complexité des commandes utilisées sur un intervalle de temps donné.
    - Pourcentage de commandes connues par l'utilisateur.
    - Pourcentage de commandes utilisées par l'utilisateur.
- Métriques reposant sur les processus cognitifs
    - Baisse du temps de réflexion sur un intervalle de temps donné.
    - Analyse des Ondes Alpha et des Ondes Beta pendant l'utilisation du logiciel.
    - Changement dans la taille des découpages mentaux lors de l'accomplissement d'une tâche donnée au cours du temps.
    - Résultats des questionnaires sur la représentation mentale du logiciel, avant et après formation.
- Métriques reposant sur les retours utilisateur
    - Commentaires liés à la notion d'*apprenabilité*.
    - Réponses à des questionnaires sur la notion d'*apprenabilité*.
- Métriques reposant sur l'utilisation de la documentation

    - Baisse du nombre de commandes d'aide utilisées sur un intervalle de temps donné.
    - Temps passé à lire la documentation avant de commencer une tâche.
    - Temps passé à accomplir une tâche après avoir lu la documentation associée.
- Métriques reposant sur les changements dans l'utilisation du logiciel
    - Comparer la qualité d'utilisation au cours du temps.
    - Comparer la qualité d'utilisation entre un utilisateur novice et un utilisateur expert.
- Métriques reposant sur des règles particulières
    - Nombre de règles requises pour décrire le système.

Ainsi, il est possible d'évaluer l'*apprenabilité* du logiciel concerné en utilisant tout ou partie des métriques présentées ci-dessus. Une première forme d'évaluation (dite *formatrice*) consistera alors à mettre en évidence les difficultés et les problèmes qui lui sont liés. Une deuxième forme d'évaluation (dite *sommative*) permettra d'établir une appréciation générale sur le système global, toujours vis-à-vis de son *apprenabilité*. Au final, cela permet d'identifier différents aspects du logiciel sur lesquels il est nécessaire d'insister au travers de différentes formes de tutoriels afin d'améliorer les conditions générales d'utilisation du système. Du point de vue utilisateur, le caractère d'*apprenabilité* d'un logiciel et les métriques pour le mesurer peuvent alors être utilisés pour lui fournir un retour sur ses performances générales dans l'utilisation du logiciel et sur les fonctionnalités avec lesquelles il doit se perfectionner.

Dans le cadre de notre étude, l'outil que nous souhaitons développer vise autant à former des novices qu'à accompagner des personnes plus expérimentées pour compléter leurs connaissances. Ainsi, nous nous intéresserons aussi bien à l'*apprenabilité initiale* qu'à l'*apprenabilité prolongée* des procédures métier concernées, en considérant un public possédant une expérience variée. Bien que le système dans lequel s'effectuent ces procédures ne soit pas purement logiciel, nous nous appuierons sur certaines des métriques présentées précédemment durant la conception de notre outil de formation. En particulier, les métriques reposant sur les performances liées à l'accomplissement de tâches semblent parfaitement adaptées. En effet, elles nous paraissent facilement transposables en des éléments ludiques tels que décrits dans la partie 1.2.3.2 (score lié au temps passé pour accomplir la procédure ou encore au nombre d'erreurs effectuées).

1.3.2 Plusieurs formes de tutoriels de logiciel

Afin de combler la complexité grandissante des logiciels et donc leur caractère *d'apprenabilité* en baisse, une solution possible est l'utilisation de tutoriels de logiciel. Nous nous intéressons dans cette partie à décrire les différentes formes de tutoriels qui existent aujourd'hui ainsi que leurs caractéristiques respectives.

1.3.2.1 Les tutoriels passifs

Les tutoriels ***passifs*** représentent les premiers tutoriels de logiciel qui sont apparus afin d'assister les utilisateurs dans leur manipulation de l'application. Dans le cadre de cette étude, nous appelons tutoriel *passif* toute forme d'aide qui n'interagit pas directement avec le logiciel ciblé. Ce sont donc principalement des documentations textuelles et/ou imagées (Tuck & Olsen, 1990) (Grabler, Agrawala, Li, Dontcheva, & Igarashi, 2009) (Carroll, 1990), ou encore outils animés comme par

exemple des vidéos (Grossman & Fitzmaurice, 2010)(Palmiter & Elkerton, 1991). L'utilisation de ce type d'assistance présente plusieurs problématiques, dont la cause principale est le caractère passif qui leur est inhérent. Comme évoqué régulièrement au cours de ce chapitre, cela ne favorise pas l'apprentissage des fonctionnalités du logiciel concerné en décourageant et désintéressant parfois les apprenants. En effet, lire une documentation qui regroupe un ensemble d'aides diverses et variées n'est pas toujours particulièrement intéressant ni adapté pour des utilisateurs qui rencontrent souvent des problèmes précis liés à une tâche en particulier. Dans le cas de longue documentation, il peut même être difficile pour ces utilisateurs de trouver l'aide recherchée initialement. En ce qui concerne les aides animées, les utilisateurs sont forcés de travailler au même rythme que l'animation (lecture de la vidéo par exemple), ce qui n'est pas toujours évident et peut donc amplifier ce phénomène de découragement et de désintéressement.

Afin d'améliorer cette première forme de tutoriels, sans malgré tout en retirer leur propriété passive, plusieurs recherches ont été menées. Les travaux de Carroll (Carroll, 1990) (Farkas & Williams, 1990) constituent une des principales études qui ont permis de faire évoluer les tutoriels *passifs* dans les années 1990, en introduisant la notion de *documentation minimaliste*. Cette notion vient s'opposer à la notion de *documentations système*, où des explications générales sur les différentes fonctionnalités du logiciel sont fournies à l'utilisateur via des leçons triviales. Selon Carroll, de telles documentations ne permettent pas aux utilisateurs de mettre en perspective les connaissances acquises dans le contexte de tâches plus complexes, et finalement échouent à enseigner de manière efficace les possibilités offertes par le logiciel. Un exemple de ce genre de tutoriel dans notre contexte industriel pourrait être une documentation qui présente les différentes actions possibles dans le logiciel de diagnostic via les différents boutons de l'interface. Si la documentation n'illustre pas l'ensemble de ces actions dans une ou plusieurs procédures réelles, le garagiste aura des difficultés à percevoir l'utilité des fonctionnalités présentées, et donc à les retenir. Afin de remédier à cette problématique, la notion de *documentation minimaliste* repose sur plusieurs principes, parmi lesquels nous pouvons citer : une ***formation mise en contexte autour de tâches réelles***, la ***modularité***, et la ***gestion des erreurs***. L'objectif est ainsi de permettre à l'apprenant de se diriger facilement vers les leçons qui le préoccupent, et de lui proposer des tutoriels sur des procédures réelles plus ou moins complexes qui aborderont plusieurs grandes fonctionnalités du logiciel. La documentation devra prévoir les erreurs que l'utilisateur pourrait effectuer, et dans ce cas lui indiquer les différentes manières de les corriger. En respectant ces principes, Carroll indique que la motivation des utilisateurs s'en trouve augmentée.

Une autre possibilité d'amélioration pour les tutoriels de logiciel passifs est mise en évidence par (Grossman & Fitzmaurice, 2010) avec l'***assistance contextualisée***. Il s'agit ici de fournir une documentation à l'utilisateur en fonction de la situation où il se trouve. L'idée proposée est alors d'améliorer le système d'*info-bulles* qui existe dans une large majorité des Interfaces Homme-Machine actuelles. En passant la souris par-dessus un élément de l'IHM concernée, une *info-bulle* apparait. L'utilisateur peut alors accéder à de la documentation textuelle et/ou une vidéo indiquant comment se servir de la fonctionnalité en question. Les résultats de ces travaux indiquent que cette méthode a permis de surmonter certains problèmes d'*apprenabilité* et d'améliorer de manière significative l'assimilation des différentes fonctionnalités du logiciel cible par les utilisateurs.

1.3.2.2 Les tutoriels actifs

La pertinence et l'efficacité des tutoriels de logiciels *passifs* sont limitées par leur nature même. Une autre catégorie de tutoriels, que nous qualifierons d'*actif*, permet alors de passer outre cette

limitation. Dans cette étude, nous appelons tutoriel de logiciel *actif* toute forme d'assistance en forte interaction avec logiciel dont il est la cible. A la différence des tutoriels *passifs*, l'objectif est ici est de proposer à l'utilisateur une aide qui évoluera automatiquement en fonction de son avancement dans une tâche donnée. Cela permet ainsi de développer le caractère *actif* de l'apprentissage en demandant à l'utilisateur d'accomplir un certain nombre d'actions avant de pouvoir passer à la suite de la procédure / du tutoriel. L'engagement des apprenants et donc l'efficacité des leçons s'en voient de cette manière largement améliorés (Li, Grossman, & Fitzmaurice, 2012).

Dans cette catégorie de tutoriels, nous pouvons citer plusieurs travaux intéressants. Dans un premier temps, nous trouvons les systèmes qui présentent les tutoriels de manière relativement traditionnelle, c'est-à-dire en utilisant du texte éventuellement combiné avec des images. Ils ressemblent donc aux tutoriels *passifs* au niveau de la forme ; la différence non négligeable est qu'ils sont capables de suivre la progression de l'utilisateur (vis-à-vis des actions effectuées dans le logiciel). Ces systèmes permettent donc de mettre à jour automatiquement l'assistance requise à un instant précis de la tâche enseignée, sans avoir besoin d'afficher la documentation entière. C'est le cas par exemple dans les travaux de (Contreras & Saiz, 1996), (Bergman, Castelli, Lau, & Oblinger, 2005), (Fernquist, Grossman, & Fitzmaurice, 2011), ou encore (Ramesh, Hsu, Agrawala, & Hartmann, 2011). Dans un second temps, il y a les systèmes qui présentent les tutoriels de manière plus visuelle et/ou dynamique. Par exemple, Pause-and-Play (Pongnumkul, et al., 2011) s'appuie sur des vidéos afin d'assister les utilisateurs. La différence majeure ici avec les tutoriels *passifs* est que le système est capable de détecter les évènements importants dans ces vidéos (ex : clic sur un bouton) et de les mettre en relation avec ces mêmes évènements dans l'application réelle concernée. Ainsi, le système est capable de mettre en pause la vidéo automatiquement si l'apprenant est trop en retard par rapport à la procédure, ou encore s'il effectue un trop grand nombre d'erreurs par rapport au contexte actuel. Dans CACTUS (Garcia, 2000), les tutoriels sont présentés sous la forme de cahier qui fait la métaphore avec le contexte scolaire dans lequel est utilisé le système. Au fur et à mesure de la leçon, les pages se tournent afin de fournir les aides correspondant au contexte courant. Les élèves ont ainsi l'impression de parcourir un livre interactif qui leur demande de réaliser des actions avant de pouvoir progresser dans l'histoire. Enfin, le système Stencils (Kelleher & Pausch, 2005) vient intégrer le tutoriel directement *par-dessus* le logiciel concerné. L'interface est recouverte par un pochoir translucide et, à chaque étape de la procédure, le pochoir ne laisse disponible à l'utilisateur uniquement l'élément IHM qui permet de progresser dans la tâche. Un ou plusieurs post-it sont aussi ajoutés sur le pochoir afin d'apporter des informations complémentaires quant à l'action que doit réaliser l'utilisateur dans le contexte actuel. De cette manière, l'apprenant est entièrement guidé directement dans le logiciel tout au long du tutoriel. Il n'a alors pas la possibilité de faire d'erreurs, et la formation se fait de manière plus rapide.

Que ce soit pour les systèmes de tutoriel *actifs* possédant une forme traditionnelle ou ceux ayant une forme un peu plus attrayante visuellement, les évaluations menées dans le cadre des travaux présentés précédemment montrent que cette méthode de formation est plus pertinente et efficace que par l'utilisation d'une simple documentation ou vidéo. Nous pouvons alors dégager de ces travaux plusieurs caractéristiques qui permettent de concevoir des systèmes de tutoriel *actifs* appropriés à nos besoins en proposant une *expérience utilisateur optimale* (Csikszentmihalyi, 1988) (Chen, Wigand, & Nilan, 1999). On remarque très rapidement que les grands principes de la *documentation minimaliste* introduits dans la partie précédente au travers

des travaux de Carroll (Carroll, 1990) en font partie. En effet, la première caractéristique qui apparait très clairement est que ces tutoriels sont créés autour de tâches ou de ***procédures réelles***. Cela assure alors la ***modularité*** du système en définissant un ensemble de leçons qui s'intéressent chacune à des procédures différentes et qui permettent d'aborder différentes fonctionnalités du logiciel. On note aussi que les tutoriels ainsi définis s'appuient sur les ***actions utilisateurs*** pour fournir l'assistance requise vis-à-vis du ***contexte courant*** de la leçon. Plus généralement, cette caractéristique permet de fournir des ***feedbacks immédiats*** à l'utilisateur, en cas d'erreur par exemple. Finalement, cette technique de formation *active* oblige l'utilisateur à effectuer les actions indiquées par le tutoriel, et pas seulement à lire les différentes aides fournies, ce qui lui permet de mieux assimiler les notions et fonctionnalités mises à disposition dans le logiciel.

1.3.2.3 Les tutoriels gamifiés

Dans une volonté de rendre les tutoriels de logiciel toujours plus intéressant et engageant pour les utilisateurs, les scientifiques ont continué à chercher des techniques innovantes. L'une de ces techniques est la ***Gamification***. Ce terme réfère à l'utilisation d'éléments de conception caractéristiques des jeux dans un contexte non-ludique (Deterding, Dixon, Khaled, & Nacke, 2011) (Deterding, Sicart, Nacke, O'Hara, & Dixon, 2011). Il ne faut pas confondre la *Gamification* avec les outils de Jeux Sérieux dans le sens où ces derniers sont conçus dès le départ pour mélanger de manière homogène une dimension ludique et une dimension sérieuse pour traiter d'une problématique spécifique (voir 1.2.1.1). Au contraire, on parle de *Gamification* lorsque l'on apporte des éléments ludiques dans un système qui n'y est pas destiné au départ. Quoi qu'il en soit, les domaines dans lesquels est utilisée la *Gamification* sont aussi variés que pour les Jeux Sérieux, tel qu'en atteste l'étude menée par (Hamari, Koivisto, & Sarsa, 2014) : commerce, éducation, santé ou encore industrie.

En particulier, le domaine qui nous intéresse est celui des systèmes de tutoriel de logiciel. Nous avons recensé plusieurs exemples de tels tutoriels *gamifiés*, aussi bien pour une application commerciale que pour une application académique dans le cadre de travaux de recherches. Dans la première catégorie, nous citerons Ribbon Hero[10] de Microsoft et Adobe LevelUp[11] (qui n'est plus disponible depuis le 30 Juin 2014). Dans ces deux exemples, un plugin a été développé afin de présenter sous forme ludique un ensemble de *missions* réparties en plusieurs *niveaux*. Chaque *mission* représente en fait un tutoriel pour apprendre à effectuer une procédure particulière dans le logiciel (ex : mettre en forme un texte dans Word, appliquer un effet de flou sur une image dans Photoshop, …). Après avoir sélectionné une *mission*, l'utilisateur est alors guidé à l'aide d'indications textuelles et/ou imagées, ou encore de contraintes de navigation dans le logiciel, afin d'achever la procédure étudiée. Une fois la *mission* accomplie, l'apprenant se voit attribuer un score en fonction du nombre d'erreurs effectuées ou du temps total passé pour terminer la procédure. Ce score est alors utilisé de plusieurs manières ludiques, comme par exemple afin de débloquer le *niveau* suivant et toutes ses *mission*s s'il est assez élevé, ou encore en le partageant sur un *leaderboard* (tableaux des scores en français, désigne un tableau qui présente les résultats des participant d'une même compétition). Dans Ribbon Hero, l'aspect ludique survient aussi dans la présentation des *niveaux* et des *missions* associées qui se fait avec une analogie de voyage dans le temps (chaque nouvelle époque permettant d'aborder des *missions* plus complexes que les

[10] https://www.microsoft.com/en-us/download/details.aspx?id=26531

[11] http://blogs.adobe.com/educationleaders/2013/06/levelup-for-photoshop-updated-for-cc.html

précédentes). Cependant, durant la *mission* en elle-même, le tutoriel prend la forme d'un système *actif* classique (voir1.3.2.2), et la dimension ludique disparait.

Dans le domaine de la recherche, on peut alors trouver plusieurs travaux qui visent à *gamifier* les tutoriels de logiciel dans leur globalité. Par exemple, les travaux de (Dong, et al., 2012) ont mené au développement de Jigsaw, une extension d'Adobe Photoshop qui propose un ensemble de puzzles à résoudre permettant à terme d'apprendre aux utilisateurs à utiliser des fonctionnalités particulières du logiciel (ex : transformation d'images en termes de position et rotation, ajustement des couleurs d'une image). La métaphore du puzzle même fait que la dimension ludique est présente à chaque instant du tutoriel, depuis la sélection du puzzle jusqu'à l'attribution des points après l'avoir résolu. De leur côté, Li et al proposent deux systèmes de tutoriels *gamifiés* pour le logiciel AutoCAD (logiciel de Conception Assistée par Ordinateur). Dans GamiCAD (Li, Grossman, & Fitzmaurice, 2012), la construction d'un vaisseau spatial fait office de contexte au tutoriel. L'utilisateur doit ainsi effectuer 5 *missions* découpées en plusieurs *niveaux* (termes inversés par rapport à Ribbon Hero et Adobe LevelUp, mais le principe est le même. Au début, l'élève a seulement accès à la première *mission* ; afin de débloquer les suivantes, il devra obtenir un score minimal de 4 étoiles sur 5 à chaque *niveau* la composant. L'attribution de score se base ici sur la vitesse d'exécution de la tâche associée au *niveau* courant, le nombre d'indices utilisés, ainsi que le nombre d'opérations d'annulation effectuées. De plus, à la fin de certaines *missions*, l'utilisateur aura la possibilité d'effectuer des mini-jeux, comme par exemple une phase d'arcade similaire au jeu Tétris[12] ou une phase où il faut relier les points pour former un objet plus ou moins complexe. L'ensemble de ces éléments de jeu ainsi que le contexte de mission spatiale assure la présence permanente d'une dimension ludique pendant le tutoriel. Dans CADament (Li, Grossman, & Fitzmaurice, 2014), la *gamification* se traduit par l'implémentation d'un environnement multijoueur où plusieurs élèves devront "s'affronter" à travers différentes tâches à accomplir le plus rapidement possible. En plus de cet aspect compétition, les grands concepts des jeux compétitifs en ligne, tels que la recherche de match ou les tableaux de récapitulation de la partie, sont repris afin de renforcer la dimension ludique.

En plus de respecter l'ensemble des caractéristiques des systèmes de tutoriel *actifs*, les tutoriels de logiciel *gamifiés* doivent donc aussi présenter un ensemble d'éléments issus du monde vidéoludique. On observe alors que toutes les caractéristiques de la dimension ludique des Jeux Sérieux sont toutes aussi pertinentes lors de la conception de ces systèmes. Ainsi, ***fantaisie***, ***score***, ***objectifs*** ou encore ***challenge*** sont autant d'éléments à prendre en compte pour obtenir un outil de formation ludique et pertinent (pour une liste complète de ces éléments, le lecteur pourra se reporter à la partie 1.2.3). Malgré un manque de preuve évident permettant d'affirmer que ces systèmes *gamifiés* favorisent un meilleur apprentissage chez l'utilisateur (Linehan, Kirman, Lawson, & Chan, 2011), les évaluations menées pour chacune des études présentée précédemment ainsi que les travaux de (Hamari, Koivisto, & Sarsa, 2014) permettent d'affirmer que la *gamification* a un effet positif sur la motivation et l'engagement de l'utilisateur lorsqu'il doit se former à un logiciel particulier (Shneiderman, 2004).

[12] http://tetris.com/

1.4 Synthèse

Dans l'objectif de concevoir un environnement de formation permettant de simuler des procédures métier qui nécessitent d'interagir avec le monde réel et des applications logicielles, nous nous sommes intéressés dans ce chapitre à établir un état de l'art des différents outils de formation existants. En particulier, nous avons initialement porté notre intérêt sur les Jeux Sérieux car ce sont des outils en vogue depuis plusieurs années qui proposent aux utilisateurs de se former de manière ludique. Ces outils combinent ainsi avec cohérence et pertinence une dimension pédagogique et une dimension ludique, chacune étant définie par différents attributs :

1. Dimension pédagogique.
   - Objectifs Pédagogiques : qualification des différents types de connaissances acquises durant la leçon.
   - Retours Pédagogiques : Feedback, Résumé.
2. Dimension ludique
   - Environnement Virtuel immersif et engageant : identification, fantaisie, liberté, mécaniques de jeu.
   - Eléments ludiques issus du jeu vidéo : scénario, objectif général et activités d'apprentissage, stimuli, score et récompenses, défi.

Dans un second temps, du fait que nous souhaitions aussi intégrer dans l'environnement de formation les logiciels nécessaires à la réalisation des procédures cibles, nous avons dirigé notre étude vers les tutoriels de logiciel. Nous avons alors constaté que la notion d'apprenabilité des logiciels est un sujet de recherche important depuis le début des Interfaces Homme Machine. Nous avons ensuite dégagé trois formes de tutoriel permettant de combler les difficultés d'apprenabilité de ces logiciels :

1. Les systèmes passifs, dont les caractéristiques importantes sont :
   - Une formation contextualisée autour de procédures réelles (à mettre en contraste avec les documentations qui décrivent les fonctionnalités du logiciel sans les illustrer au travers d'un cas d'utilisation réel et complexe).
   - La modularité.
   - La prise en compte des erreurs.
2. Les systèmes actifs, caractérisés par :
   - Les attributs des systèmes passifs.
   - Une assistance dynamique et contextualisée en fonction des actions de l'apprenant.
   - Retours pédagogiques immédiats (en cas d'erreur par exemple).
3. Les systèmes gamifiés, décrits par :
   - Les caractéristiques des systèmes actifs.
   - L'apport d'éléments de jeu dans un contexte initialement non-ludique (se référer à la dimension ludique des Jeux Sérieux pour une liste de ces éléments).

Pour le reste de ce manuscrit, l'outil de formation qui nous intéressera sera donc un environnement qui allie à la fois les caractéristiques d'un Jeu Sérieux et celles des tutoriels de logiciel gamifiés. En effet, bien qu'il nous soit possible de concevoir exhaustivement l'environnement de Jeu Sérieux dans lequel l'utilisateur se formera aux procédures dans le monde réel, il nous faut aussi prendre en compte les procédures liées aux manipulations dans une ou plusieurs applications logicielles.

Dans cette optique, nous devrons aussi prendre en compte les éléments caractéristiques des tutoriels de logiciel gamifiés, qui représentent la forme de tutoriel de logiciel la plus proche des Jeux Sérieux au travers de leur dimension ludique commune. L'une des problématiques sera alors d'homogénéiser les éléments de ces deux catégories d'outil de formation afin d'obtenir un environnement de formation final cohérent.

Chapitre 2. Méthodes et environnements pour la création assistée d'outils de formation interactifs

2.1 Motivations

Comme présenté en introduction, le deuxième axe de recherche de nos travaux a pour objectif de proposer une méthodologie globale permettant aux formateurs de modéliser formellement des leçons adaptées à l'outil de formation que nous aurons défini à partir du 0. Dans cette optique, nous avons choisi d'effectuer un tour d'horizon des outils existant permettant d'assister différents profils d'utilisateur dans le développement des différents aspects des Jeux Sérieux et des tutoriels de logiciel. Afin de compléter la littérature liée spécifiquement à l'étude de tels outils, nous avons choisi d'étendre nos recherches aux différentes méthodes facilitant la création des Jeux Vidéos, qui présentent par nature de fortes similarités avec les Jeux Sérieux (voir 1.2). Tout au long de ce chapitre, nous établirons donc la liste des différents concepts et caractéristiques sur lesquels nous nous sommes appuyés afin de définir par la suite notre propre méthodologie (Chapitre 4). Une première piste nous conduira alors vers l'étude d'outils permettant de décrire formellement et de manière plus ou moins adaptée différentes dimensions des outils de formation qui nous intéressent. Puis, une seconde piste nous permettra d'aborder l'étude de plusieurs méthodes visant à faciliter le processus de création global de tels outils.

Ainsi, nous avons tout d'abord souhaité étudier les outils existant pour décrire de manière formelle différents aspects des Jeux Sérieux, Jeux Vidéos, et tutoriels de logiciel. La plupart du temps, ces outils seront disponibles sous la forme de représentations (ou langages) textuelles (Ishida, 2002) (Devillers & Donikian, 2003) et/ou graphiques (Van Est, Poelman, & Bidarra, 2011), chacune étant plus ou moins adaptée à différents profils d'utilisateurs. Dans la partie 2.2, nous nous intéresserons donc à analyser ces différentes représentations, en mettant en évidence la ou les dimensions qu'elles permettent de décrire, ainsi que leurs points forts et leurs points faibles respectifs vis-à-vis de notre problématique initiale. Cela nous permettra alors de dégager les caractéristiques intéressantes que nous pourrons prendre en compte lors de la définition de notre méthodologie afin que les formateurs puissent décrire formellement et de manière adaptée les leçons qu'ils souhaitent enseigner au travers de notre outil de formation.

Dans un second temps, nous avons souhaité étudier des méthodes et environnements informatiques dont l'objectif général serait d'apporter des outils visant à améliorer le processus global de création d'outils de formation. Dans la partie 2.3, nous nous intéressons donc à extraire différentes fonctionnalités permettant de réduire la charge de travail des utilisateurs et de les assister tout au long de leur tâche. Une première grande famille de méthodes que nous étudierons concerne des techniques de génération automatisée (Araújo & Roque, 2009) (Grabler, Agrawala, Li, Dontcheva, & Igarashi, 2009) de certaines dimensions des tutoriels de logiciel (que nous aurons préalablement établies dans la partie 2.2). De manière complémentaire, une deuxième grande famille de méthodes étudiées concernera l'utilisation d'environnements auteur (Torrente, Moreno-Ger, Fernández-Manjón, & Sierra, 2008) (Resnick, et al., 2009) afin de simplifier les tâches devant être effectuées manuellement.

2.2 Des représentations pour décrire formellement différentes composantes

Afin de décrire formellement les différentes composantes (activités pédagogiques (Garcia, 2000), évolution des objets et acteurs dans l'environnement virtuel (Ishida, 2002) (Devillers & Donikian, 2003), gameplay (Van Est, Poelman, & Bidarra, 2011)) des outils de formation étudiés, différentes représentations existent. Dans cette partie, nous cherchons ainsi à étudier les différentes formes que peuvent prendre ces représentations, afin d'en extraire les caractéristiques intéressantes qui pourront éventuellement être réutilisées dans la méthodologie que nous proposerons par la suite. Pour cela, nous ferons un tour d'horizon des différents formats de représentation existants, qu'ils soient textuels (2.2.1) ou graphiques (2.2.2), et analyserons leurs particularités respectives afin d'en dresser les points forts et les points faibles.

2.2.1 Représentations textuelles

La première famille de représentation existante afin de décrire les différentes dimensions des outils de formation étudiés jusqu'à présent concerne les notations (ou langages) que nous appelons « textuelles ». Cette partie s'attache à décrire cette première famille de représentation, en prenant soin de bien faire la distinction entre les langages existants (2.2.1.1), c'est-à-dire ceux étant communément reconnus au niveau industriel ou scientifique, et les langages dédiés (2.2.1.2), c'est-à-dire ceux ayant été créés pour répondre à un besoin spécifique.

2.2.1.1 Langages et notations existantes

Comme pour toute application logicielle, la manière la plus directe afin de créer des Jeux Sérieux (ou plus généralement des Jeux Vidéos) ainsi que des tutoriels de logiciel est de les développer à partir de zéro, à l'aide de langages de programmation dis « bas-niveaux » (comme par exemple le langage C++[13] ou le langage Java[14]). Bien que cette méthode soit tout à fait acceptable, elle sort du contexte de nos travaux où nous nous intéressons uniquement aux méthodes impliquant une description partielle des outils de formation considérés. Dans cette partie, notre objectif est ainsi de présenter des langages (ou notations) qui sont (ou ont été) utilisés afin de décrire certaines composantes de ces outils.

Nous pouvons tout d'abord relever les langages utilisés au travers de Moteurs de Jeux, qui sont des environnements informatiques permettant initialement de créer des Jeux Vidéos, et donc par extension des Jeux Sérieux, à l'aide de plusieurs outils mis à disposition de l'utilisateur. Des langages de programmation répandus constituent une partie de ces outils, et peuvent être utilisés afin de personnaliser le comportement de chaque élément du jeu considéré. Unity[15], Unreal Engine[16], et Cry Engine[17] sont aujourd'hui les moteurs de jeux commercialisés les plus connus et donc les plus utilisés par les équipes de développements[18]. L'idée fondamentale sur laquelle

[13] http://www.stroustrup.com/C++.html

[14] https://docs.oracle.com/javase/8/

[15] https://unity3d.com/

[16] https://www.unrealengine.com/

[17] https://www.cryengine.com/

[18] http://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/

repose ces trois environnements et les langages de programmation associés, est le fait que l'univers de chaque jeu est constitué d'*objets*, eux-mêmes constitués de plusieurs *composants*. Chacun de ces composants permet alors de définir une partie du comportement de l'objet associé, afin par exemple de décrire les règles de collisions avec les autres objets ou encore de préciser la manière dont cet objet doit être affiché à l'écran. A l'aide de langages de programmation (C# et JavaScript principalement pour Unity, C++ pour Unreal Engine et Cry Engine), l'utilisateur est alors en mesure de décrire ses propres composants et ainsi de personnaliser ses jeux. Cependant, l'utilisation de tels langages se fait de manière brute et restent donc en dehors de la portée des formateurs.

Un autre exemple de langage utilisé afin de personnaliser différentes composantes de Jeux vidéo est le langage Ruby[19], sur lequel repose les différentes versions du Moteur de Jeux RPG Maker[20]. Cet outil met à disposition un ensemble de fichiers qui se suffisent à eux-mêmes afin de créer un jeu de rôle fonctionnel et toutes les caractéristiques qui le composent. Cependant, l'utilisateur est libre de modifier ces fichiers afin de personnaliser ces différentes caractéristiques : menus, combats, déplacements, etc… Cela requière malheureusement de fortes compétences en programmation, puisqu'il s'agit ici de savoir examiner un comportement existant, pour ensuite le modifier et l'adapter à ses propres besoins.

Les langages extensibles tels qu'XML[21] représentent une autre famille de notation existante utilisée pour décrire diverses situations. En particulier, (Katsurada, Nakamura, Yamada, & Nitta, 2003) présentent dans leur article le langage XISL, qui est une extension du langage XML. Ce langage permet de décrire des scénarios d'interactions multimodales s'exécutant dans des navigateurs web. Il permet ainsi de contrôler des flux et des transitions de dialogues, en prenant en considération les entrées / sorties liés aux différents navigateurs web. Bien que le contexte de cet article ne corresponde pas directement à nos travaux, il est intéressant d'observer qu'il est possible d'utiliser un langage textuel –tel qu'XML– plus accessible que des langages de programmations plus brutes afin de décrire et de modéliser des scénarios (ici, des enchainements de dialogues dynamiques).

Finalement, l'utilisation de tels langages textuels ne nous apparait pas comme pertinente dans le cadre de notre étude. En effet, en plus de leur nature textuelle non adaptée à des personnes sans expertise particulière en informatique et en programmation, ces notations ont été conçues afin de pouvoir être utilisées dans des situations toutes plus variées les unes que les autres pour développer un large panel d'applications logicielles. Ces langages sont ainsi intrinsèquement complexes à maitriser, et il est donc peu envisageable pour nous de les proposer directement à des formateurs.

2.2.1.2 Langages dédiés

Tout en restant dans la même logique de décrire textuellement certaines composantes des outils de formation étudiés, des personnes ont cherché à proposer des langages qui soient conçu dans l'unique but de décrire une composante particulière. Des langages ainsi définis sont appelés

[19] https://www.ruby-lang.org

[20] http://www.rpgmakerweb.com/

[21] https://www.w3.org/XML/

langage dédiés (DSL[22] en anglais), et proposent des fonctionnalités adaptées à l'objectif pour lequel ils ont été conçus, ce qui leur permet d'être plus accessibles.

Dans la littérature, il est possible de trouver des exemples de tels langages dédiés. Dans l'article de (Ishida, 2002), l'auteur introduit le langage Q, qui permet de décrire des interactions entre les agents (ou acteurs) qui appartiennent à une scène. Ce langage représente une interface entre les programmeurs professionnels et les personnes qui écrivent des scénarios de jeu. Q dérive du langage Schème ; ainsi, il est possible d'utiliser les différentes structures qu'il propose, mais il implémente aussi de nouvelles commandes afin de décrire des situations particulières. De plus, le langage Q apporte la notion de *signal* et d'*action*, qui corresponde respectivement à l'observation et la modification de l'environnement dans lequel se déroule le scénario. Cela permet d'implémenter un système d'évènements qui soit relativement accessible. Malgré tout, décrire un scénario avec ce langage reste relativement bas-niveau et n'est pas tout à fait à la portée d'une personne ne maitrisant pas la programmation. Dans cette optique, un processus particulier a été mis en place afin de concevoir efficacement ces scénarios :

- Dans un premier temps, un « écrivain » et un programmeur vont définir simultanément une version du scénario en langage naturel, et les différentes fonctionnalités qui seront utilisées et devront être accessibles au travers du langage Q.
- Dans un second temps, l'« écrivain » va décrire le scénario à l'aide du langage Q. Cela est maintenant possible étant donné que ses diverses fonctionnalités ont été définies auparavant en le faisant intervenir directement. Dans le même temps, le développeur implémente les diverses fonctionnalités
- Optionnellement, un troisième intervenant peut définir des patrons d'interaction pour ce scénario, sous la forme de "carte de patron". Cette carte peut être remplie à l'aide d'un tableur comme Excel, pour ensuite être traduite en Q par un compilateur / interpréteur. Cela permet par la suite de faciliter le travail de l'écrivain.

On observe dans cet article qu'il est possible de proposer à des personnes non-informaticiennes des notations textuelles qui soient plus adaptées que des langages classiques, en leur proposant uniquement les fonctionnalités dont ils ont besoin pour un scénario donné. Cela implique alors de mettre en place un processus bien particulier, ce qui peut représenter une contrainte forte dans certains cas (coopération entre l'écrivain et le programmeur pas toujours possible). Nous retiendrons cependant les notions de *signal* et d'*action*, qui nous sont apparues très intéressantes dans le cadre de nos travaux. En effet, nous pouvons mettre ces deux concepts en relation avec notre besoin qui est de pouvoir décrire des leçons se déroulant dans un environnement informatique, nécessitant des actions de la part de l'utilisateur d'une part (= *signal*), et l'apport d'informations en fonction des actions (= *action*).

Dans l'article de (Devillers & Donikian, 2003), nous avons pu observer un deuxième exemple de langage dédié permettant de décrire des scénarios d'orchestration d'acteurs au sein d'un monde virtuel en 3D. Pour cela, le langage s'appuie sur le principe des machines à état ; ainsi, toutes les spécificités du langage sont transposables en machines à état, et vice-versa. L'auteur présente dans son article la grammaire de son langage, qui donne la possibilité pour les utilisateurs de décrire les scénarios, les acteurs, et l'ordonnancement des scénarios. Le langage introduit donne aussi la possibilité d'intégrer à tout moment des instructions C++, afin de pouvoir exprimer

[22] Domain Specific Language

certains besoins non autorisés par la grammaire du langage. Au terme de la description du scénario, celui-ci est entièrement compilé vers le langage C++. Malheureusement, il reste assez bas-niveau (très proche des langages de programmation évoqués en 2.2.1.1) à utiliser et est donc destiné à des programmeurs. L'approche par machine à état est cependant très intéressante car elle permet de représenter des scénarios de jeu assez simplement. Malgré tout, plus les scénarios se complexifient, plus la modélisation en machine à état devient complexe et s'adresse à des programmeurs ou des informaticiens.

D'un point de vue plus commerciale, les propriétaires de GameMaker : Studio[23] proposent avec leur outil le langage dédié GML[24] (GameMaker Language), destiné à apporter plus de flexibilité et de contrôle dans le développement des jeux, par rapport aux actions standard proposées au travers de l'environnement auteur et de ses différentes interfaces. En particulier, il est possible d'utiliser ce langage afin :

- de définir des scripts. Ces scripts servent principalement à implémenter des fonctions personnalisées qui pourront plus tard être appelées pendant le développement du jeu.
- de réagir à des évènements. En plus des comportements prédéfinis et accessibles directement dans GameMaker, il est possible de définir des comportements personnalisés à l'aide de GML afin de contrôler l'évolution des objets du monde virtuel.
- de réagir à la création d'une "pièce". Dans GameMaker, une pièce représente une scène où le joueur devra effectuer des actions. Une pièce peut donc par exemple correspondre à un niveau du jeu, où le joueur devra contrôler son personnage afin de remplir un certain nombre d'objectif, mais aussi au menu principal où le joueur pourra choisir de créer une nouvelle partie de charger une partie existante. A l'aide de GML, il est ainsi possible de définir pour chaque pièce des instructions personnalisées qui seront exécutées à chaque fois que celle-ci devient active.
- de réagir à la création d'une instance d'objet. Dans GameMaker, il est possible de définir des objets avec des comportements communs. Afin d'ajouter effectivement ces objets dans le jeu, il est ensuite nécessaire de les instancier et de les lier aux pièces du jeu. A l'aide de GML, il est alors possible de définir des comportements spécifiques pour chaque instance d'un même objet, qui seront exécutés à leur création.

Ainsi, le langage GML dispose d'une grammaire propre et de plusieurs fonctionnalités associées afin d'offrir à l'utilisateur plus de contrôle lors de la création de ses jeux. Il reste malgré tout assez proche d'un "langage classique", et il nous est de ce fait difficile d'imaginer utiliser un tel langage dans le cadre de nos travaux. Nous retiendrons cependant les notions d'*objet* et d'*instance d'objet*, qui nous paraissent pertinentes afin par exemple de définir des "classes" d'instructions pouvant être personnalisées à chaque fois qu'elles sont instanciées lors de la définition des leçons à enseigner.

Finalement, au travers des exemples présentés ci-dessus, nous avons pu relever certains concepts intéressants que nous pourrions réutiliser dans nos travaux. Cela est principalement dû au fait que chacun de ces langages dédiés ait été conçus pour répondre à des besoins spécifiques, et qu'ils ne sont donc pas destinés à être utilisés dans des contextes plus globaux au contraire des langages de programmations "classiques". Cependant, nous ne sommes pas convaincus de

[23] http://www.yoyogames.com/gamemaker

[24] https://docs.yoyogames.com/source/dadiospice/002_reference/001_gml%20language%20overview/

l'accessibilité de ces langages, qui restent difficiles d'accès pour un non-programmeur du fait de leur nature textuelle.

2.2.2 Représentations graphiques

En opposition avec les représentations textuelles se trouvent les représentations que nous appelons « graphiques ». Cette deuxième famille présente des avantages ergonomiques évidents que nous nous efforcerons de mettre en évidence tout au long de cette partie. De la même manière que pour les représentations textuelles, nous décrirons dans un premier temps les représentations graphiques existantes (2.2.2.1), puis nous analyserons dans un second temps celles ayant été définies dans le cadre de travaux spécifiques (2.2.2.2).

2.2.2.1 Représentations existantes

Au fil du temps, plusieurs représentations graphiques ont fait leur apparition afin de décrire de différentes manières le comportement de différents systèmes. Dans cette optique, on peut ainsi relever UML, les Flowcharts, les Réseaux de Pétri, BPMN, ou encore IDEF (Tang, et al., 2008) (Araújo & Roque, 2009). Tout au long de cette partie, nous intéressons à étudier des exemples du monde scientifique qui reprennent certaines de ces notations afin de modéliser différentes composantes des Jeux Vidéos ou des Jeux Sérieux.

Dans leur article, (Tang, et al., 2008) décrivent un modèle de définition de scenario adapté aux Jeux Sérieux, ainsi que les avantages qui lui sont associés. Après avoir défini les différents composants des Jeux Sérieux à modéliser (interfaces utilisateurs, monde virtuel et objets, scénario), les auteurs explicitent les critères que devra respecter le langage de modélisation adopté. En particulier, il devra permettre de représenter et de décrire l'ensemble des éléments évoqués précédemment, tout en restant simple d'utilisation. Après avoir parcouru l'ensemble des langages de modélisation et avoir dressé les avantages et les inconvénients, ils concluent qu'UML celui étant le plus adapté à leurs besoins. En effet, malgré sa complexité, c'est celui qui est le plus largement utilisé dans l'ingénierie logicielle du fait de la multitude de systèmes/composants informatiques qu'il permet de représenter avec ses différents types de diagramme. Les auteurs proposent alors différentes améliorations qu'ils ont choisi d'apporter aux diagrammes UML afin de pouvoir combler différents manques du langage. En particulier, les diagrammes d'états, utilisés afin de décrire les scénarios de Jeux Sérieux, se voient enrichis de notations additionnelles afin de pouvoir représenter des séquences d'*acte* (un acte étant la description d'une action par ou sur un objet du monde virtuel). Vis-à-vis de nos recherches, l'utilisation d'UML et plus spécifiquement des diagrammes d'état est donc une piste intéressante car la modélisation d'une leçon peut s'apparenter à la modélisation d'un scénario tel que décrit dans cet article.

De leur côté, (Araújo & Roque, 2009) présentent dans leur article comment il est possible de modéliser des scenarios de jeu en ayant recours aux réseaux de Pétri. Ce choix est justifié par plusieurs arguments à l'encontre des deux alternatives envisagées :

- UML : comme expliqué précédemment, c'est un langage difficile à maitriser et donc à utiliser, à cause justement de la variété de diagrammes mis à disposition. De plus, il est sujet à plusieurs interprétations différentes car il manque de consistance et de formalisation sémantique.

- les Flowcharts. Bien qu'ils permettent de modéliser des processus à l'aide de nœuds conditionnels et de transitions liant ces nœuds, on ne peut les utiliser que pour des systèmes séquentiels non parallèles. C'est d'ailleurs une des raisons pour lesquelles les Flowcharts sont rarement utilisés dans le domaine du Jeu Vidéo.

De leur côté, les réseaux de Pétri sont utiles pour représenter graphiquement et mathématiquement les différents états d'un système discret. Ils sont de puissants outils d'analyse, et permettent de vérifier un grand nombre de propriétés d'un système. Les auteurs présentent alors plus précisément les réseaux de Pétri, qui se basent principalement sur le trio Jetons-Places-Transitions, puis en montrant différentes techniques "avancées" qui permettent de modéliser différents types de transitions ou conditions logiques. Afin d'illustrer l'usage d'une telle représentation, les auteurs donnent plusieurs exemples concrets et plus ou moins complexes. Ces exemples mettent alors en évidence les différents avantages des réseaux de Pétri : les notations utilisées sont simples, il est possible de décrire des hiérarchies de diagrammes afin de modéliser à plusieurs niveaux de détail un même système, et ils présentent des propriétés très intéressantes afin de valider et de simuler les scénarios modélisés. A l'inverse, il est possible de noter la complexité de description et de lecture grandissante de cette représentation à mesure que le scénario évolue ou que l'on veut préciser certains points.

Nous retiendrons des réseaux de Pétri qu'ils sont un outil puissant dans la modélisation de scénarios de jeu, du fait du peu de notation qu'ils utilisent ainsi que de leur caractère mathématique et multiprocessus / parallèle. Cependant, à cause du peu de notation qu'ils proposent, il leur manque une "signification graphique" claire vis-à-vis des différents éléments et comportements modélisés. Il est ainsi difficile de distinguer différentes classes d'action (comme par exemple une action effectuée sur la voiture et une autre effectuée sur le logiciel). Cela fait qu'ils sont donc assez complexes à utiliser et à lire dans le cas de scénarios de grande taille, et ne sont donc pas à la portée de toute personne non-informaticienne.

Enfin, (Panzoli, et al., 2014) présentent dans leur article une méthodologie de conception de Serious Games multijoueur collaboratifs, au travers de leur propre réalisation, 3DVOR, qui est un Serious Game multijoueur collaboratif autour d'une pièce d'opération virtuelle. Les auteurs y développent une méthodologie qui leur permet de respecter certaines contraintes : édition de scénario, conception d'environnement virtuel multijoueur, et intégration d'Intelligence Artificielle. En particulier, la première étape consiste à modéliser les différentes activités de chaque personnage du scénario. Pour cela, ils ont choisi d'utiliser BPMN (Business Process Modeling Notation), un standard permettant de modéliser des flux de processus (White, 2004). BPMN a été conçu afin de fournir une notation universelle pouvant être lue et comprise par un grand nombre de personnes, en particulier par tout utilisateur professionnel. Cette notation peut être utilisée afin de modéliser des processus d'activités qui interagissent entre-elles, ou encore afin de modéliser des processus internes, à plusieurs niveaux de précisions. Le langage est constitué d'éléments graphiques le plus différenciable possible, et en nombre relativement peu élevé afin d'être facile à retenir, à lire, et à utiliser. On y trouve d'abord les objets de flux :

- Les Evènements, représentées sous la forme d'un cercle. Ce sont des éléments qui permettent d'agir sur des flux et/ou des activités.
- Les Activités, représentées par des rectangles. Ils permettent de définir l'exécution de différentes tâches / activités.

- Les Portes, représentées par des losanges. Ils permettent de contrôler la divergence et la convergence de flux de séquences d'activités.

Dans le cadre de 3DVOR, ces objets de flux sont utilisés pour décrire les différentes actions devant être réalisées afin d'effectuer une activité, et sont complétés par des conditions afin de décrire formellement les changements impliqués dans l'Environnement Virtuel. On trouve ensuite plusieurs types de flèches de flux permettant principalement de décrire les séquences entre les objets de flux. Il est aussi possible de séparer les séquences d'activités ou les processus dans des Couloirs, eux-mêmes pouvant se subdiviser avec des Lignes. Les Couloirs sont utilisés lorsque le diagramme présente plusieurs entités ou participants, et permettent d'organiser et de catégoriser des activités au sein d'un même processus grâce aux lignes. Au final, BPMN représente un langage graphique très intéressant dans le contexte de notre étude, puisqu'il introduit des notations ayant initialement été conçues afin d'être utilisable et compréhensible par différentes catégories de personnes, incluant des gens n'ayant pas ou peu d'expertise en informatique / programmation, ce qui correspond à notre besoin de départ.

Finalement, nous avons pu étudier dans cette partie différentes représentations graphiques existantes et standardisées pouvant être adaptées aux besoins de nos travaux afin de décrire des leçons de formation. Malgré leurs avantages et leurs inconvénients respectifs, ces langages correspondent, à différents niveaux, à nos attentes concernant leur facilité de prise en main par des personnes non-informaticiennes.

2.2.2.2 Représentations personnalisées

En dehors des représentations graphiques existantes, certaines personnes ont cherché à proposer de nouvelles représentations personnalisées afin de répondre à des besoins bien spécifiques. Ces représentations étant donc fortement liées à un contexte industriel ou scientifique précis, elles sont très souvent accompagnées de divers processus ou d'environnements connexes. Dans cette partie, nous décrivons donc plusieurs de ces environnements qui introduisent des représentations graphiques personnalisées.

Tout d'abord, nous pouvons relever les travaux notables de (Van Est, Poelman, & Bidarra, 2011). Dans leur article, les auteurs présentent un Environnement Auteur permettant d'éditer des scénarios de jeux autour d'un contexte donné. Les auteurs distinguent deux catégories d'environnements auteur liés à la création de Jeux Vidéo / Jeux Sérieux : ceux qui permettent l'édition d'Environnement et ceux qui permettent l'édition de scénario Causal. Dans les premiers, l'utilisateur crée de toute pièce l'environnement du jeu qui sera ensuite présenté au joueur (ex : Unity, Unreal Engine, RPG Maker, …). Avec les deuxièmes, les utilisateurs éditent le ou les processus associés à un scénario de jeu, sous forme de graphe la plupart du temps. Ces scénarios sont appelés Causal car il permette de décrire les actions devant se dérouler en fonction des agissements du joueur, par exemple : "quand l'utilisateur clique sur ce bouton, j'affiche ce message". De plus, la principale difficulté rencontrée avec les outils auteurs de Jeux Sérieux actuels est qu'ils n'ont pas été conçus explicitement pour être utilisé par des personnes non expertes en informatique ou en conception de jeu. Ainsi, en partant du principe qu'il est plus simple d'éditer un graphe qu'un environnement de jeu complet, les auteurs se sont intéressés dans leurs travaux à proposer des outils permettant à des personnes non informaticiennes mais expertes dans un ou plusieurs domaines métier de définir des scénarios modaux plus ou moins complexes.

Afin de répondre à cette problématique, qui est très similaire à la nôtre, les auteurs introduisent alors dans l'article trois niveaux d'éditions :

- Le premier est le niveau bas, le niveau "programmation" (utilisation de langages textuels tels que C++).
- Le deuxième niveau, intermédiaire, est le niveau de définition du Gameplay ; c'est à ce niveau que les Game Designers (concepteurs de jeu) interviennent généralement, en utilisant des fonctions décrites au niveau bas pour définir les interactions du joueur avec le jeu.
- Les auteurs apportent un 3ème niveau : le niveau d'édition de scénario. A ce niveau, on se concentre sur le scénario en lui-même, et non sur les contraintes de gameplay ou de programmation, c'est un niveau d'abstraction supplémentaire.

Afin de décrire ces scénarios, les auteurs proposent l'utilisation de "briques", qui sont divisées en un certain nombre de types (actions, évènements, logique, …). Chaque brique possède un certain nombre de propriétés qu'il est possible de configurer afin de s'adapter à différentes situations (par exemple, une brique "le personnage marche jusqu'à la voiture" peut posséder une propriété "vitesse de marche"). Chacune des briques doivent ensuite être liées entre-elles par des transitions indiquant l'ordre dans lequel devront s'exécuter les actions associées. L'environnement auteur présenté par la suite permet à l'utilisateur de manipuler ces différentes briques à l'aide d'interactions simples. Au final, cet article introduit un bon nombre de pistes intéressantes dans le cadre de nos travaux. En particulier, le concept de brique graphique configurable nous parait vraiment pertinent du fait qu'elle constitue en soi un élément simple à manipuler et donc accessible à des non-informaticiens. Cela permet aussi de proposer un ensemble de briques fixe et de taille raisonnable afin que l'utilisateur puisse les assimiler rapidement.

De leur côté, les environnements Scratch (Resnick, et al., 2009) et Alice (Kelleher & Pausch, 2007) ont pour vocation de faciliter l'apprentissage de la programmation pour les enfants, en leur proposant des outils leur permettant de créer des projets de jeu en 2D ou en 3D de manière relativement simple. L'accessibilité de ces deux environnements s'est construite autour de la méthodologie adoptée afin de laisser les utilisateurs implémenter des programmes (qui définissent le comportement des différents objets ajoutés dans le jeu) à la manière des Légos ou de puzzles. Ainsi, les utilisateurs peuvent choisir des *blocs* dans une liste prédéfinie, chacun ayant sa propre fonctionnalité (structure de contrôle, déplacement de personnage, création de variable, etc.), pour ensuite les assembler dans un éditeur visuel. Chaque bloc possède une forme qui indique explicitement à l'utilisateur comment il peut être assemblé avec d'autre bloc. De plus, certains blocs proposent un ou plusieurs menus déroulants afin de spécifier leurs éventuelles propriétés configurables, alors que d'autres permettent d'imbriquer à l'intérieur (et non à la suite) d'autres blocs, telles que des conditions. Tout est présenté de manière lisible, et tous les détails techniques plus ou moins complexes liés à la syntaxe que l'on retrouve dans les langages de programmation habituels sont présentés graphiquement de façon à ce qu'ils deviennent quasiment transparents pour l'utilisateur.

Le concept de représentation graphique assimilable à l'assemblage de Légos ou de pièces de puzzles introduit par les deux environnements Scratch et Alice est particulièrement intéressant. En effet, cela permet de décrire des comportements plus ou moins complexes pour chaque objet présent dans le jeu de manière très visuelle et accessible à un très large public. De plus, cela permet de s'assurer de la bonne syntaxe des instructions utilisées, et ainsi d'obtenir à la fin un

scénario fonctionnel (dont seul le déroulement pourra ne pas correspondre aux attentes initiales). Cependant, les blocs disponibles pour décrire les comportements sont volontairement laissés en grand nombre, et certains proposent des actions unitaires assez bas-niveau. Cela a été fait ainsi afin que les utilisateurs puissent personnaliser le plus possible leurs jeux, mais cela rajoute une certaine complexité qui demande un temps non négligeable afin de s'approprier l'ensemble des fonctionnalités mises à disposition.

Dans un autre contexte, (Barot, Lourdeaux, Burkhardt, Amokrane, & Lenne, 2013) introduisent dans leurs travaux le langage graphique ACTIVITY-DL, qui permet de décrire sous forme d'arbre hiérarchique les tâches devant être effectuées par un opérateur dans un environnement virtuel associé. Chaque tâche et sous-tâche ainsi décrite sont directement associées à des éléments du modèle définissant l'ensemble des éléments requis afin de reproduire les activités concernées dans l'environnement virtuel. Cela permet de n'utiliser qu'un unique type d'entité graphique (sous forme de bloc rectangulaire) afin de représenter une large variété de tâches. De plus, les relations entre ces tâches et sous-tâches se représentent simplement par des transitions entre les blocs associés, et la description de l'enchainement de ces tâches se fait à l'aide d'un ensemble d'opérateurs bien définis. Avec une telle représentation, l'objectif des auteurs est alors de permettre une spécification formelle assistée des procédures métier décrites par des experts du domaine.

Enfin, dans un autre registre, nous pouvons citer les représentations graphiques introduites dans certains moteurs de jeux commerciaux. La plus notable est liée au concept de *Blueprint* (schéma, en anglais) proposé par l'environnement Unreal Engine. Alors que ce dernier permet à des profils de développeur de définir des éléments de Gameplay à l'aide du langage C++ (voir 2.2.1.1), il propose aussi un éditeur graphique permettant à des concepteurs d'assembler ces éléments de Gameplay afin de définir les règles qui régissent l'évolution du jeu. La représentation graphique associée est très proche du concept des briques graphiques configurables introduis précédemment au travers de l'article de (Van Est, Poelman, & Bidarra, 2011). L'utilisateur peut alors connecter des nœuds, des évènements, des fonctions ou encore des variables (tous présentés sous la forme de blocs ou de propriétés d'un bloc) avec des transitions afin d'implémenter des comportements et fonctionnalités plus ou moins complexes. Bien sûr, Unreal Engine étant un outil très puissant permettant de créer des jeux dits "AAA" (c'est-à-dire des jeux impliquant des moyens de production et des équipes de développement importants), la prise en main de l'ensemble des fonctionnalités mises à disposition par le système de *Blueprint* demande un certain degré d'implication des personnes qui seront amenés à l'utiliser. Cependant, les concepts graphiques de base qui lui sont liés et sa vocation initiale (définir un niveau de description du Gameplay des jeux accessibles à des personnes sans compétence en programmation) viennent renforcer notre idée d'envisager un langage graphique pour permettre à des formateurs de décrire formellement les leçons qu'ils souhaitent créer.

Finalement, cette partie nous a permis d'introduire des représentations graphiques, et surtout des concepts qui leur sont associés, ayant été conçus dans le but de proposer des outils adaptés à des profils d'utilisateur variés ne possédant pas nécessairement de compétences en programmation ou en conception information. Bien que le niveau d'accessibilité de ces représentations dépende directement de la variété de situations qu'elles permettent de décrire, nous relevons une tendance forte commune à chacune d'elles. Il y a en effet une volonté de proposer à chaque fois une bibliothèque (finie ou qu'il est possible d'enrichir au fil du temps) d'entités graphiques (blocs, briques), chacune permettant de modéliser des comportements bien définis que l'utilisateur peut

assimiler rapidement. De plus, afin de limiter la taille de cette bibliothèque et donc de faciliter sa prise en main générale, chacune de ces entités peut posséder des propriétés configurables afin de pouvoir réutilisée dans différentes situations. Les différentes séquences d'actions qui se dérouleront dans le jeu seront alors définies par l'assemblage de ces entités graphiques (transitions, imbrication).

2.3 Des outils et des méthodes au service de l'utilisateur

En complément des représentations permettant de décrire formellement certaines composantes des outils de formations étudiés, il peut être intéressant de proposer d'autres outils afin d'accompagner leur utilisateur et de lui simplifier la tâche. Comme nous avons pu l'observer en 2.2, des environnements auteurs sont souvent une première étape vers cet objectif. Dans cette partie, nous nous intéressons alors à étudier des outils et des méthodes qu'il peut être intéressant de proposer au travers d'un environnement auteur afin d'assister le plus possible l'utilisateur. Dans un premier temps, nous nous intéresserons donc à des outils de générations automatisés ayant été proposés au fil du temps dans le monde scientifique (2.3.1). Puis, nous étudierons diverses méthodes permettant d'améliorer le processus global de création d'outils de formation (2.3.2).

2.3.1 Outils de génération automatisée

Afin d'accompagner le formateur dans sa tâche de description formelle d'une leçon, une idée qui apparait rapidement intéressante est la faculté de pouvoir générer automatiquement ou semi automatiquement tout ou partie de cette description. Au cours de nos recherches, nous avons pu observer que d'autres personnes avant nous s'étaient intéressées à cette problématique dans le cadre de la génération de tutoriels de logiciel. Il nous est apparu que les outils de génération automatisés alors introduits pouvaient être classifiés en deux grandes catégories. La première, que nous étudierons en 2.3.1.1, concerne les outils permettant de générer une suite d'instructions à partir du résultat attendu. De son côté, la deuxième catégorie, qui sera l'objet de la partie 2.3.1.2, regroupe les outils utilisant la technique de *Programmation par Démonstration*.

2.3.1.1 Génération à partir du résultat

Une première approche ayant pour but de générer de manière automatisée des tutoriels consiste à analyser le résultat souhaité afin d'en déduire le processus permettant d'arriver à ce résultat. On peut trouver dans la littérature quelques travaux qui illustrent cette approche et l'appliquent à différents domaines.

C'est le cas par exemple de (Li, Zhang, & Fitzmaurice, 2013) qui décrivent dans leur article une méthode de génération automatique de tutoriaux autour du logiciel AutoCAD, et plus particulièrement sur comment reproduire un dessin à bases de formes simples telles que des lignes, des cercles, etc. et de contraintes entre ces formes. Le principe sur lequel se base leur système est lié à la recherche de la séquence d'actions optimale (appeler *Plan*) permettant de reconstruire un dessin donné en entrée. Dans un premier temps, un premier système (le préprocesseur) va prendre en entrée une image (un fichier AutoCAD), qu'il va traiter afin de définir l'ensemble des objets et des contraintes qui le composent. Ensuite, un deuxième système (le *Solver*) va traiter le problème et fournit l'ensemble des actions permettant de reconstruire l'image d'entrée. Pour cela, un ensemble de règles permettant de contraindre le problème et de

trouver la meilleure solution est défini en suivant le Keystroke Level Model (John & Kieras, 1996) (KLM). Puis, le problème est résolu à l'aide d'un système ASP (Answer Set Programming) spécifique : DLV. Finalement, un dernier système va prendre en entrée ce *Plan* d'actions et le transformer en un tutorial étape par étape, en associant à chaque action une aide visuelle à base de texte ou d'images (qui pourront être générées à partir de captures d'écran).

De leur côté, (Harms, Cosgrove, Gray, & Kelleher, 2013) proposent une méthode afin de générer des tutoriaux similaires à ceux présentés en 1.3.2.2 via le système Stencil (Kelleher & Pausch, 2005). Ces tutoriaux ont pour objectif d'enseigner aux utilisateurs de l'environnement Looking Glass les étapes nécessaires afin de créer des histoires animées et les différents programmes associés permettant de spécifier le comportement de tous ses composants (environnement 3D, acteurs, animations, etc.). Pour cela, les tutoriaux utilisent une mise en page particulière s'appuyant sur le principe de pochoir permettant de contrôler les éléments d'interface avec lesquels peut interagir l'utilisateur. Au travers de leur système, l'utilisateur peut choisir de charger des projets existants disponibles via la communauté en ligne de Looking Glass. Il peut ensuite choisir les différents morceaux de code sur lesquels il souhaite obtenir un tutorial. A partir de là, l'architecture dirigée par les modèles implémentée par Looking Glass leur permet d'isoler le ou les éléments d'interfaces permettant d'instancier chaque instruction du morceau de code choisit, et ainsi de générer l'aide associée. De plus, grâce à un système de dépendance entre les différents modèles de Looking Glass, le système est capable de générer dynamiquement les éventuellement étapes intermédiaires avant de pouvoir effectuer l'étape en elle-même. Par exemple, si l'utilisateur doit ajouter une instruction de type "boucle" et que l'onglet actuellement sélectionné dans Looking Glass n'est pas celui où se trouve cette instruction de "boucle", une étape intermédiaire sera automatiquement créée afin d'indiquer à l'utilisateur qu'il doit d'abord sélectionner le bon onglet. Au contraire, si l'onglet actuellement sélectionné est bien celui où l'instruction de "boucle" se trouve, cette étape ne sera pas créée.

Enfin, (O'Rourke, Andersen, Gulwani, & Popović, 2015) présentent dans leur article un système permettant de générer des tutoriaux interactifs afin d'enseigner étape par étape comment résoudre divers problèmes mathématiques dont la solution peut s'exprimer à l'aide d'un algorithme. Ainsi, il est par exemple possible de générer un tutoriel pour enseigner la manière d'effectuer une soustraction. Pour cela, la première étape est de décrire l'algorithme permettant de résoudre le problème en question à l'aide d'un langage particulier, le TPL (Thought Process Language). TPL permet d'utiliser des primitives communes à beaucoup de langages, telles que la déclaration de variable, l'utilisation d'opérateurs entiers (addition, soustraction, etc…) ou booléens (égalité, infériorité, …), ou encore l'utilisation d'instructions d'assignement, conditionnelles, et de boucles. Une fois l'algorithme décrit avec ce langage, il est ensuite compilé puis interprété ligne par ligne afin de produire divers types d'évènement (ex : création de variable, exécution d'une boucle, etc.). Chacun de ces évènements est alors traité par des fonctions spécifiques (*interface hooks*) dont l'implémentation dépend de l'application sur laquelle s'exécute le tutorial, et dont l'objectif est de fournir une explication visuelle adapté au type d'évènement. Ces explications peuvent alors être apportées :

- de manière textuelle. Pour cela, le système recherche dans une liste de modèles de phrases celui qui correspond au contenu de la ligne traitée. Ce modèle est alors complété en utilisant le nom des variables ou des fonctions utilisées dans cette ligne.

- en mettant en surbrillance des éléments d'interface afin de contextualiser les explications textuelles (ex : mise en surbrillance du premier digit de chaque nombre pour expliquer une opération de soustraction).

Finalement, nous avons pu observer dans cette partie différentes approches partageant un objectif commun : générer des tutoriels de manière automatique en se basant sur certaines données d'entrée. L'avantage de tous ces systèmes est que les tutoriels sont générés dynamiquement et permettent donc aux utilisateurs d'obtenir l'aide qu'ils souhaitent quand ils le souhaitent en fonction de la tâche qu'ils souhaitent accomplir. En revanche, ils nécessitent d'avoir un ensemble de règles, pouvant prendre plusieurs formes (problèmes ASP, dépendances de modèles, algorithmes de résolution, etc.), qui permettent d'extraire à partir d'un contexte particulier les différentes étapes permettant de parvenir à ce même résultat. Cela peut alors devenir problématique dans le cas où il n'est pas possible d'exprimer le problème sous forme de règles, ou encore s'il existe plusieurs manières d'effectuer une même tâche et que l'on souhaite insister sur une manière particulière.

2.3.1.2 Programmation par démonstration

Une seconde approche possible permettant d'automatiser la génération de tutoriels interactifs est d'adopter la technique assez répandue de programmation par démonstration. Le principe de cette méthode consiste à observer les actions effectuées par un utilisateur dans un environnement informatique (un logiciel la plupart du temps), pour ensuite les formaliser à l'aide de langages textuels ou graphiques. Dans cette partie, nous nous intéressons ainsi à étudier les travaux de plusieurs chercheurs qui ont choisi d'adapter cette technique afin de faciliter la création de tutoriels.

Par exemple, le système Teach me While I Work (TWIW) (Contreras & Saiz, 1996) qui permet d'exécuter des tutoriels actifs sur certaines applications (voir partie 1.3.2.2), est complété par un environnement d'édition permettant de créer et de modifier ces tutoriels. Les logiciels ciblés peuvent être représentés sous forme d'arbre décrivant l'ensemble des tâches et des sous-tâches qu'ils permettent de réaliser. Il est alors possible de distinguer deux types de tâches : les tâches *atomiques* et les tâches *composées*. Les tâches *atomiques* sont les feuilles de l'arbre, et correspondent à une interaction *atomique* de l'utilisateur avec le logiciel. Au contraire, les tâches *composées* représentent les nœuds de l'arbre, et sont donc décrites par plusieurs tâches *atomiques* et/ou *composées*. Chaque tâche *atomique* est définie par un certain nombre d'informations, tels que les éléments d'interface correspondants ou encore les pré/post actions associées. Ainsi, l'environnement auteur permet simplement de définir ces tâches ainsi que leur hiérarchie. Pour les tâches *composées*, l'auteur doit simplement spécifier le nom de cette tâche. Pour les tâches *atomiques*, l'auteur interagit directement avec le logiciel ciblé ; l'atelier capture alors l'évènement correspondant et l'associe à la tâche souhaitée en utilisant la technique de programmation par démonstration.

Dans l'article de (Garcia, 2000), une technique similaire est utilisée afin de générer les cours sous formes de cahier de l'environnement CACTUS (voir partie 1.3.2.2) pour enseigner l'utilisation de plusieurs logiciels tels que des environnements de programmations ou des applications pour créer les plannings scolaires. Dans leur forme finale, ces tutoriaux sont définis via un langage de programmation qui n'est donc pas accessible un non-initié. C'est pourquoi CACTUS propose un mode d'édition et une méthodologie particulière afin de permettre aux

concepteurs de formation de créer leurs propres tutoriaux interactifs ou d'en modifier des existants. Cette méthodologie repose ainsi sur la technique de programmation par démonstration, qui leur permet d'*observer* les actions effectuées par le formateur dans le logiciel cible, pour ensuite générer les activités pédagogiques associées.

Du côté des applications de traitement d'image (Photoshop / GIMP), (Grabler, Agrawala, Li, Dontcheva, & Igarashi, 2009) se sont intéressés dans leur étude à développer un outil aussi basé sur la technique de programmation par démonstration afin de générer des tutoriaux sur la manipulation de photographies / d'images. Pour cela, un premier système (le *Demo Recorder*) va enregistrer la succession de changements d'état associés aux opérations effectuées sur une image donnée. Ces changements d'états sont divisés en trois grandes familles (communes à la plupart des applications de manipulation d'image) : la sélection de l'opération à effectuer, l'édition des paramètres liés à l'opération sélectionnée, et enfin l'application de l'opération sélectionnée. Ces différents changements d'état vont ensuite être regroupés afin de réduire au maximum le nombre d'étapes dans le tutoriel (ex : si plusieurs opérations de changements de saturation se succèdent, elles sont regroupées en une seule opération prenant en compte uniquement la saturation initiale de l'image et la saturation finale). Puis, un deuxième système (l'*Image Labeler*) va rechercher et marquer les différentes zones importantes de l'image à l'aide d'algorithmes spécifiques. De cette manière, il est possible de détecter les différentes régions des environnements extérieurs (ciel, terre, …) et les différentes régions du visage présents dans l'image, afin de décrire plus précisément certaines étapes du tutoriel qui exigent d'effectuer des opérations locales sur l'image. Enfin, un troisième système (le *Tutorial Generator*) va prendre en entrée les sorties des deux premiers systèmes afin de générer le tutorial étape par étape qu'il faut suivre pour obtenir le résultat final à partir de l'image initiale.

Les travaux de (Chi, et al., 2012) font suite à cette étude, avec l'implémentation de l'environnement MixT qui permet de générer des tutoriaux utilisant plusieurs types de média (images, vidéos enrichies, textes) afin d'apprendre à manipuler des logiciels d'édition d'images. Cette génération s'effectuer en trois étapes. Dans un premier temps, l'auteur va devoir effectuer la procédure ciblée par le tutoriel. Pendant cette phase, MixT enregistre plusieurs éléments : la liste des opérations effectuées dans le logiciel, une trace des évènements souris (clics, mouvements, cliqué-glissé), et une vidéo de l'ensemble de la procédure. Dans un second temps, MixT va utiliser ces différents éléments afin de procéder à un ensemble d'opérations sur la vidéo permettant de générer l'ensemble des médias qui composeront le tutoriel : segmenter clairement chaque étape de la procédure, identifier les zones de l'interface dans lesquelles s'effectue chaque étape afin d'enrichir la vidéo, et définir une capture d'écran qui identifie chaque étape de manière représentative. Enfin, MixT va combiner chacun de ces médias afin de former le tutoriel final qui sera présenté dans une interface web.

Très utilisée, la technique de génération par démonstration s'est aussi vue améliorée au cours de ces dernières années afin de répondre à certaines problématiques. Dans leurs travaux, (Ramesh, Hsu, Agrawala, & Hartmann, 2011) s'intéressent à proposer un environnement capable de transformer des tutoriels interactifs générés pour un logiciel particulier (par exemple, Adobe Photoshop) afin de les adapter semi automatiquement à un logiciel similaire (par exemple, GIMP). Cela se fait notamment au moyen de tables d'équivalence entre les modèles de chaque logiciel ciblé, qui contiennent respectivement la liste de toutes les opérations disponibles. Bien que ces tables doivent être construites manuellement, ces recherches sont très intéressantes puisqu'elles permettent d'appliquer un même tutoriel généré une fois à n'importe quel logiciel qui propose

d'effectuer les mêmes opérations. De leur côté, (Bergman, Castelli, Lau, & Oblinger, 2005) s'intéressent dans leur étude à proposer une méthode permettant d'adapter des tutoriels de logiciel existants afin qu'ils soient toujours valides lorsque le logiciel concerné évolue (ajout de nouvelles fonctionnalités, changements d'interface, etc.). Ainsi, l'environnement DocWizards est capable de générer une première fois les tutoriels en question, à l'aide de la technique de programmation par démonstration. Puis, lors d'éventuelles démonstrations futures de cette même procédure (effectuées par exemple à chaque évolution du logicielle), un algorithme d'apprentissage pourra comparer les opérations effectuées et automatiquement modifier le tutoriel initial en conséquence. Bien qu'ils exigent une intervention manuelle à chaque évolution du logicielle pour effectuer de nouveau la procédure souhaitée, ces travaux constituent des pistes intéressantes dans le cadre de notre étude puisqu'ils abordent des problématiques qui sont communes à tous les tutoriels de logiciel.

De manière générale, on peut observer au travers de ces différents travaux que la technique de programmation par démonstration est une technique assez populaire dans le monde scientifique afin d'automatiser la génération de tutoriels de logiciels. Cela est dû au fait qu'elle soit relativement simple à mettre en place, avec une procédure qui requiert uniquement l'observation et l'enregistrement des actions effectuées par un expert dans le logiciel cible. Bien que nous n'ayons pas été en mesure de trouver de travaux similaires appliqués au domaine des Jeux Vidéos / Jeux Sérieux (afin par exemple de scénariser les actions du joueur), cette technique représente une piste très intéressante que nous considèrerons lors de la définition de notre méthodologie.

2.3.2 Offrir une expérience globale adaptée aux compétences du concepteur de formation

Au travers de notre étude, nous avons pu observer qu'il était assez fréquent d'implémenter un environnement auteur et certaines fonctionnalités (Mehm, Reuter, Göbel, & Steinmetz, Future trends in game authoring tools, 2012) afin de rendre plus accessible la création d'outils de formations. En particulier, certaines représentations et outils de génération introduits dans les parties 2.2 et 2.3.1 sont associées à de tels environnements afin d'être manipulables plus simplement. Suivant les besoins, les environnements auteurs proposent alors différentes caractéristiques, principalement liées à la mise en place d'une ergonomie adaptée. Cela passe par exemple par l'implémentation d'une interface utilisateur moderne (technologie, accessibilité des outils en fonction du contexte courant, personnalisation de la mise en page, couleurs, etc.) (Torrente, Moreno-Ger, Fernández-Manjón, & Sierra, 2008, IEEE) (Mehm, Göbel, Radke, & Steinmetz, 2009), ou par la mise en place d'un processus bien défini permettant à l'utilisateur de prendre en main efficacement les différentes opérations possibles (Mehm, 2010). Mais cela peut aussi passer par l'introduction de fonctionnalités particulières permettant de rendre l'application accessible à différents profils de personnes (étrangers, handicapés (Torrente, Vallejo-Pinto, Moreno-Ger, & Fern, 2011)). Ceci dit, les caractéristiques que nous venons de décrire étant relativement communes aux applications logicielles d'aujourd'hui, nous avons choisi dans le cadre de nos travaux de nous intéresser à des fonctionnalités plus originales qu'il serait intéressant de mettre en place dans le processus de création d'outils de formation que nous définirons par la suite.

Une de ces fonctionnalités repose dans le principe de découper le modèle de description d'un outil de formation en plusieurs couches. Ce découpage consiste à définir différents niveaux de

description, chacun possédant son propre format, son propre degré de précision, et sa propre complexité. Par exemple, (Tang, Hanneghan, & Carter, 2013) présentent dans leur article l'architecture du moteur de jeux implémenté, qui s'appuie sur 3 couches :

- le premier niveau est celui qui est qui est directement spécifié au travers de l'environnement auteur associé. Il permet de décrire le modèle des objets qui composent le jeu ainsi que les relations / interactions entre ces objets, et cela de manière graphique afin d'être adapté aux compétences de l'auteur.
- un deuxième niveau permet ensuite de représenter ce même modèle en y ajoutant la logique nécessaire pour définir l'évolution du jeu au cours du temps. La représentation utilisée pour décrire ce deuxième niveau reste indépendante de l'application qui exécutera le jeu à la fin.
- le troisième niveau permet finalement de décrire ce modèle du jeu et sa logique dans une technologie compréhensible par l'application qui exécutera le jeu au final.

Similairement, les briques graphiques configurables utilisées par (Van Est, Poelman, & Bidarra, 2011) pour décrire des scénarios de Jeux Sérieux représentent des entités à leur plus haut niveau d'abstraction. Elles sont en fait définies par un ensemble de blocs graphiques moins accessibles pour des non-experts en informatique, eux-mêmes définis par un ensemble d'instructions utilisant un langage de programmation bas-niveau. L'environnement de modélisation de scénarios pédagogiques collaboratifs présentés par (Ferraris, Lejeune, Vignollet, & David, 2005) présentent aussi une architecture générale divisée en plusieurs couches afin de permettre aux formateurs de décrire leurs propres scénarios. Un tel découpage en couches est intéressant puisqu'il permet de rendre transparentes des problématiques purement informatiques pour des utilisateurs non-initiés. Cela se fait cependant au détriment d'une précision amoindrie du fait que l'utilisateur ne peut décrire que des objets, des comportements, ou plus généralement des entités, au préalablement prévus, ce qui limite donc ses possibilités.

La deuxième grande famille de fonctionnalité que nous avons pu observer au travers de notre étude concerne la création et l'utilisation de "patrons". C'est un concept dont l'objectif est de mettre à disposition des utilisateurs finaux un ensemble d'objets complexes prédéfinis qui puissent être ajoutés dans leurs jeux et personnalisés simplement en fonction de leur besoin. C'est le cas de (Mehm, Göbel, & Steinmetz, 2011) qui présentent dans leurs travaux un environnement auteur et une méthode permettant de modéliser des patrons de *gameplay* au travers de *composant*s. Ils utilisent pour cela l'approche de la Programmation Orientée Composant (Component-Based Software Engineering). Un composant est défini par plusieurs éléments, dont notamment :

- une interface qui regroupe les différents attributs et paramètres associés. C'est cette interface que devront ensuite spécifier les futurs auteurs afin de personnaliser le composant.
- une implémentation propre à l'atelier auteur, indépendante de la technologie finale utilisée.
- une implémentation spécifique pour chaque technologie qui sera amenée à exécuter un jeu avec le composant en question.

Notons que l'on retrouve dans ces éléments le découpage en trois couches présentées plus haut, mais cette fois-ci au niveau particulier des objets définissant un jeu. Les notions de découpage en couche et de patrons peuvent donc être intimement liées, comme par exemple avec les briques graphiques introduites par (Van Est, Poelman, & Bidarra, 2011) qui regroupent

finalement d'autres briques graphiques afin de définir un comportement général facilement réutilisable. Plus généralement, le principe de patron est assez courant dans l'industrie du jeu vidéo. Par exemple, le moteur de jeu Unity permet la création de *prefab*s, qui représentent un objet avec un ensemble de propriétés et de comportements pouvant être instancié autant de fois que nécessaire sans aucune action particulière. De son côté, le moteur de jeu Unreal Engine permet la création de *Blueprints* (déjà abordés dans la partie 2.2.2.2), qui en eux-mêmes sont des regroupements d'instructions C++ définissant des comportements variés. Le concept de patron est donc très intéressant puisqu'il permet de capitaliser des entités plus ou moins complexes, afin par la suite de les réutiliser directement dans différentes situations en précisant uniquement certains de leurs paramètres.

Finalement, cette partie nous a permis d'introduire, en plus des notions liées à l'ergonomie, deux grands concepts intimement liés pouvant jouer un rôle important dans la recherche d'une méthode permettant d'offrir une expérience globale de création d'outils de formation adaptée aux compétences des formateurs. En effet, en masquant des détails techniques d'implémentation et en proposant des outils visant à permettre la réutilisation d'objets (au sens large) complexes, les principes de découpage en couches et de patron représentent des idées très pertinentes que nous chercherons à adapter dans le cadre de la méthodologie que nous définirons.

2.4 Synthèse

Afin d'être en mesure de proposer une méthodologie de création d'outils de formation adaptée à des formateurs (ou plus généralement à des personnes non-expertes en informatique), nous avons effectué un état de l'art des différents outils existants (dans le monde scientifique et industriel) ayant pour vocation d'assister les utilisateurs dans cette tâche. Ainsi, nous nous sommes tout d'abord intéressés à différentes représentations permettant de décrire formellement différentes composantes des outils de formation. Nous avons alors observé deux grandes catégories de représentation :

- Les représentations textuelles, ou plus communément les langages de programmation. Nous en avons ensuite extrait les notations existantes, c'est-à-dire celles représentant un standard dans l'informatique et utilisées pour créer toute sorte d'application, et les langages dédiés, c'est-à-dire les représentations ayant été conçues pour répondre à un besoin précis et décrire des composantes spécifiques dans un contexte particulier.
- Les représentations graphiques, dont l'intérêt principal est de permettre de représenter le fonctionnement de systèmes plus ou moins complexes de manière ergonomique. Ici aussi, nous avons été menés à étudier des notations graphiques reconnues dans le monde de l'informatique, et des notations personnalisées permettant de décrire des systèmes spécifiques.

Bien que la deuxième catégorie de représentation paraisse la plus adaptée à notre problématique initiale, nous avons pu souligner tout au long de la partie 2.2 un ensemble de concepts intéressants que nous prendrons en considération lors la définition de notre méthodologie dans le Chapitre 4.

Puis, nous avons effectué un tour d'horizon de diverses méthodes ayant pour vocation de fournir aux utilisateurs des outils pour les accompagner, directement ou indirectement, tout au long du processus de création de l'outil de formation. En particulier, nous nous sommes intéressés

aux outils permettant d'automatiser la description formelle des procédures enseignées au travers des tutoriels de logiciel. Là encore, nous en avons extrait deux grandes catégories :

- ceux qui permettent de générer un ensemble d'instructions à partir du résultat souhaité. Il s'agit ici de définir un ensemble de règles permettant de déduire les étapes requises pour parvenir à ce résultat, avec les actions requises de la part de l'utilisateur dans l'interface du logiciel, et d'éventuelles aides visuelles (images) pour illustrer le tutoriel.
- ceux qui utilisent la technique de programmation par démonstration. En demandant à un expert du domaine d'effectuer la procédure étudiée sur le logiciel, l'objectif ici est alors d'observer et d'enregistrer les actions correspondantes, pour ensuite générer les instructions associées.

La deuxième catégorie d'outils de génération automatisés nous a alors semblé être la plus adaptée à nos travaux, dû au fait qu'elle soit aisément adaptable à un grand nombre d'applications logicielles. Enfin, nous avons présenté deux grands concepts (découpage en couche, patrons) liés aux environnements auteurs qu'il serait intéressant d'appliquer dans le cadre de nos travaux dans notre volonté d'améliorer le processus de création d'outils de formation.

Partie II. Contributions

Chapitre 3. Conception d'un outil de formation pour l'enseignement d'activités métier dans un environnement informatique modulaire

3.1 Vue d'ensemble

Maintenant que nous avons une vision globale du contexte dans lequel se placent nos travaux, nous allons introduire dans ce chapitre l'outil de formation qui nous permet de répondre à une de nos problématiques initiales : l'enseignement d'activités métier mettant en jeu des interactions avec le monde réel ainsi qu'avec un ou plusieurs logiciels associés. La Figure 4 montre les éléments important de cet outil. Dans un premier temps, nous définirons donc l'Environnement de Formation dans lequel l'apprenant évoluera afin de réaliser les différentes leçons proposées. En particulier, nous expliciterons les différents systèmes qui le composent et qui permettent de reproduire l'environnement de travail complet de l'élève. De plus, nous présenterons les différents éléments pédagogiques et ludiques qui pourront être intégrés à notre outil afin de garantir sa double vocation sérieuse et divertissante. Dans un second temps, nous nous intéresserons à définir les différentes composantes du *scénario*, qui représente la modélisation formelle d'une leçon dans l'Environnement de Formation et à destination des apprenants. Nous observerons alors que la définition de ces scénarios repose en partie sur le modèle implémenté par cet EF.

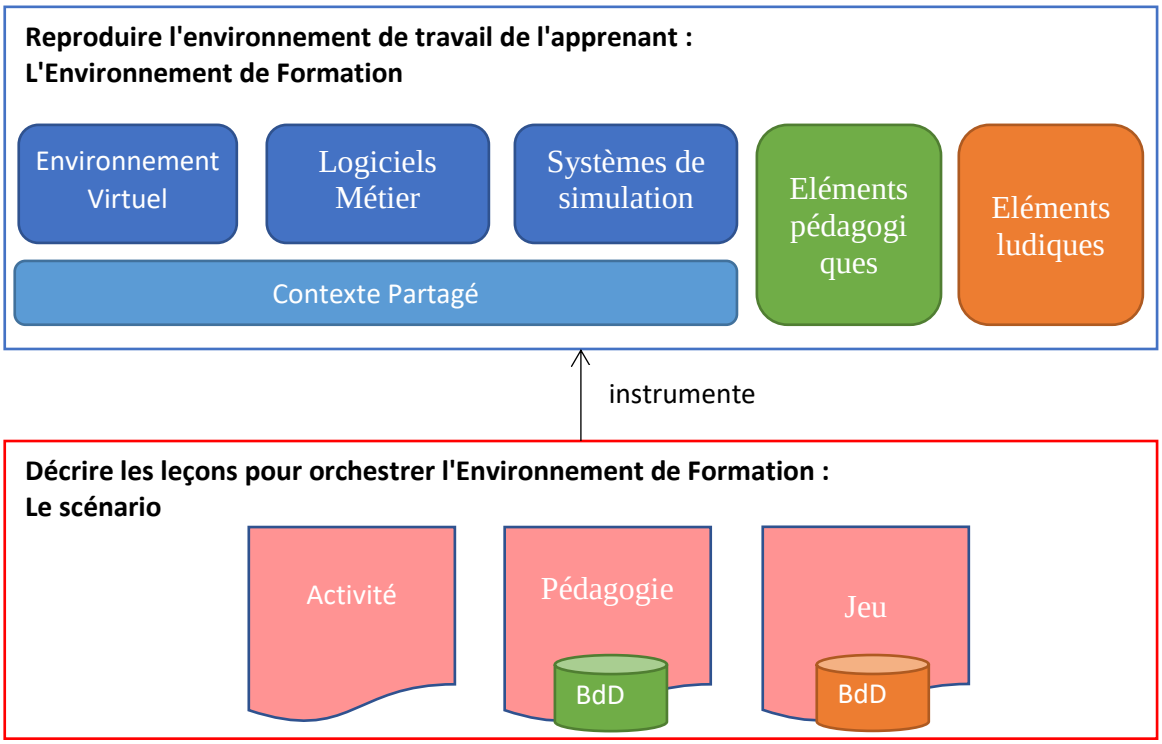


FIGURE 4. DECOMPOSITION DE L'OUTIL DE FORMATION EN SES 2 GRANDES FONCTIONNALITES

3.2 Environnement de Formation

Dans cette première partie de notre troisième chapitre, nous nous intéressons à introduire l'Environnement de Formation (EF) sur lequel nous nous appuierons pour le reste du manuscrit. Nous définissons le contexte d'utilisation de notre outil ainsi que ses différentes caractéristiques dans la partie 3.2.1. Nous détaillons ensuite en 3.2.2 les différents modules qui composent notre Environnement de Formation et qui nous permettent de reproduire l'environnement de travail de l'apprenant afin que ce dernier puisse réaliser la ou les activités souhaitées. Puis, nous introduisons le besoin d'un contexte partagé pour homogénéiser cet Environnement de Formation et présentons le modèle complet qui nous permettra d'orchestrer l'évolution des différents modules en fonction des interactions effectuées par l'apprenant. Enfin, nous présentons dans la partie 3.2.3 les éléments de notre environnement qui définissent les dimensions pédagogiques et ludiques de notre outil.

3.2.1 Introduction

Dans le 0, nous avons établi un état de l'art de différents outils de formations interactifs et ludiques pour l'enseignement d'activités métier. Nous nous sommes alors intéressés à deux formes d'outils de formation particulières, chacune étant adaptée à différents types d'activités métier. De leur côté, les Jeux Sérieux sont des outils qui conviennent afin de former des apprenants à des opérations nécessitant d'interagir avec le monde réel, puisqu'ils permettent de reproduire et de simuler dans des environnements virtuels les différents objets et interactions associés aux activités ciblées. D'un autre côté, nous nous sommes intéressés aux tutoriels de logiciel du fait que les activités étudiées impliquent aussi l'utilisation de logiciels métier. L'utilisation du logiciel réel plutôt qu'une reproduction partielle du logicielle (telle que cela aurait été fait dans un Jeux Sérieux) permet alors un meilleur transfert des connaissances pendant la session de formation.

L'Environnement de Formation (EF) qui nous intéresse dans ces travaux et qui permet de reproduire l'environnement de travail global de l'apprenant est ainsi constitué de plusieurs systèmes informatiques, que nous appelons *modules*. Chacun de ces modules est représenté dans l'EF à l'aide d'un modèle particulier, de manière à rendre générique la description de l'évolution de l'EF en regard des actions de l'apprenant, et ce quel que soit le domaine métier concerné. Ces modèles sont ensuite instanciés de manière appropriée afin de correspondre aux besoins de la leçon étudiée, et doivent être facilement compréhensible par une personne non-experte en informatique. Les deux principaux modules sont d'un côté l'Environnement Virtuel (EV) issu des Jeux Sérieux et de l'autre le ou les Logiciels Métier (LM) nécessaires à l'accomplissement des opérations souhaitées. En ce qui concerne l'Environnement Virtuel, nous avons souhaité concevoir un système qui soit valable quelle que soit la situation considérée. Le modèle que nous en proposons alors (3.2.2.1) nous permet de représenter de la manière la plus simple et la plus exhaustive possible les différents environnements de travail possibles de l'apprenant. Afin que l'implémentation de l'EV corresponde à sa représentation effective dans l'EF, ce modèle est alors directement instancié par l'EV. En ce qui concerne les Logiciels Métier, nous observerons qu'il n'est pas possible d'en définir un modèle aussi exhaustif et uniforme que celui de l'EV afin de décrire les opérations qui y ont lieu. En effet, la représentation des logiciels sera presque toujours différente du aux objectifs hétérogènes et aux différents domaines métier qu'ils servent. Le modèle permettant de décrire le module des Logiciels Métier sera alors très général (3.2.2.2). De plus, nous verrons en 3.2.2.3 que les LM utilisés ont dans certains cas besoin d'interagir

directement avec des éléments du monde réel pour fonctionner correctement. Dans le cadre de notre Environnement de Formation et dans un souci pratique, nous souhaitons nous affranchir de ces interactions. Un troisième module concernera donc les différents Systèmes de Simulation (SS) permettant aux Logiciels Métier de continuer à fonctionner correctement. Pour les mêmes raisons que les LM, nous utiliserons un modèle assez général pour représenter ces systèmes de simulation.

De plus, notre Environnement de Formation propose une expérience à la fois ludique et pédagogique, à la manière des Jeux Sérieux et des tutoriels de logiciel gamifiés. Pour cela, nous nous appuyons sur les différentes caractéristiques de ces deux formes d'environnements d'apprentissage, extraites de notre état de l'art afin de définir un système global cohérent aussi bien au niveau de la dimension ludique (3.2.3.1) qu'au niveau de la dimension pédagogique (3.2.3.2). En effet, notre Environnement de Formation mêlant ces deux types d'outil, il est important pour nous d'assurer une homogénéité dans les interactions entre l'apprenant et ces différents systèmes de formation.

La Figure 5 et la Figure 6 présentent deux captures d'écrans qui montrent un exemple de comment peut se présenter l'Environnement de Formation durant l'exécution d'une leçon.

FIGURE 5. ENVIRONNEMENT DE FORMATION : VUE ENVIRONNEMENT VIRTUEL

La première vue montre l'Environnement Virtuel où l'apprenant peut effectuer un certain nombre d'interactions sur une voiture, alors que la deuxième montre le logiciel nécessaire pour effectuer une activité de diagnostic sur un véhicule. Un point important à noter est que le module d'Environnement Virtuel et le module des Logiciels Métier doivent être accessibles à tout instant de la leçon (sauf éventuellement durant certaines phases scriptées pour placer l'élève dans une situation précise), par exemple ici à l'aide de la vue secondaire en haut à gauche sur lequel il est possible de cliquer afin d'alterner les vues. Le module concernant les systèmes de simulation est quant à lui caché à l'utilisateur puisque ceux-ci ne représentent pas des outils ou des objets avec lesquels il faudra interagir afin d'effectuer les opérations de la leçon. De plus, des informations liées à la pédagogie sont disponibles en permanence, comme par exemple ici dans un bandeau

pédagogique avec le nom de l'activité courante et un message indiquant comment progresser dans cette activité.

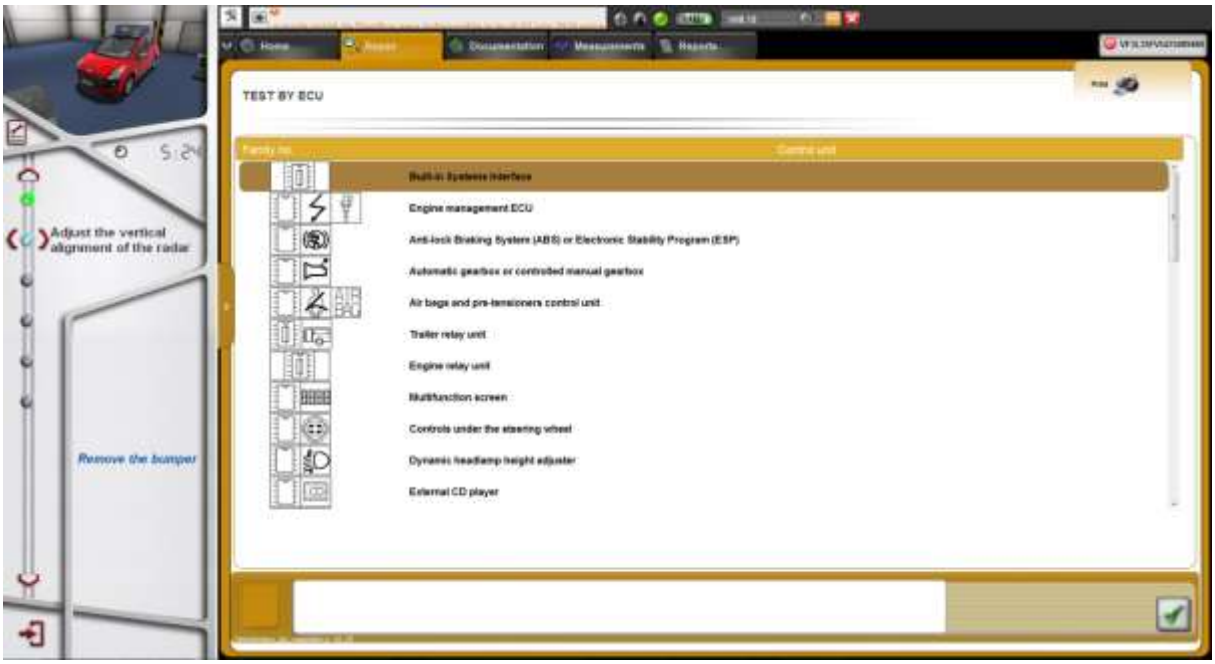


FIGURE 6. ENVIRONNEMENT DE FORMATION : VUE LOGICIEL METIER

3.2.2 Un environnement informatique modulaire

3.2.2.1 L'Environnement Virtuel

L'Environnement Virtuel constitue l'un des deux modules directement accessibles par l'utilisateur. Son objectif principal est de reproduire le plus fidèlement possible l'environnement de travail de l'apprenant, avec tous les objets qui le composent et l'ensemble des interactions associées. Par exemple, il peut servir à modéliser une salle d'opération dans le cadre d'un contexte médical, ou encore un garage de réparation dans un contexte automobile. Il participe aussi à la dimension ludique de notre Environnement de Formation, avec par exemple la possibilité pour l'élève d'évoluer plus ou moins librement dans un monde virtuel. Afin de pouvoir être utilisable dans un large panel de situations professionnelles, l'EV instancie un modèle particulier permettant de représenter de manière simple mais aussi de manière exhaustive les différents environnements de travail de l'apprenant dans des domaines métier variés. La Figure 7 présente ce modèle.

Ainsi, notre Environnement Virtuel est d'abord défini par un ensemble d'Environnements. Par exemple, nous pourrions avoir un EV constitué d'une salle d'opération et d'une salle de réveil en se plaçant dans un contexte médical. Chaque Environnement possède ensuite plusieurs Points de Vue, c'est-à-dire des positions de caméra prédéfinis permettant à l'utilisateur d'accéder à des angles de vues particuliers et intéressants de l'Environnement associé. Chaque Environnement possède aussi un certain nombre d'Objets. Un Objet peut représenter n'importe quelle entité physique présente dans l'environnement de travail habituel de l'apprenant. Par exemple, la salle d'opération va contenir un Objet "table d'opération" ainsi que divers Objets nécessaires à l'opération (scalpel, ciseaux, seringues). Un Objet peut aussi représenter une personne ou un être

vivant de manière générale, telle qu'une infirmière, un anesthésiste, ou le patient. Chaque Objet est défini par un ensemble d'États qui peuvent prendre un ensemble défini ou non de Valeurs, afin de représenter son statut courant. Par exemple l'Objet "patient" peut avoir un État "douleur" pouvant prendre les valeurs "élevée", "moyenne", ou "basse". Chaque Objet est aussi défini par un ensemble d'Interactions, elles-mêmes définies par un ensemble de Changements d'État. Par exemple, en effectuant l'Interaction "injecter morphine" de l'Objet "patient", son État "douleur" passe de sa valeur actuelle à la valeur "basse".

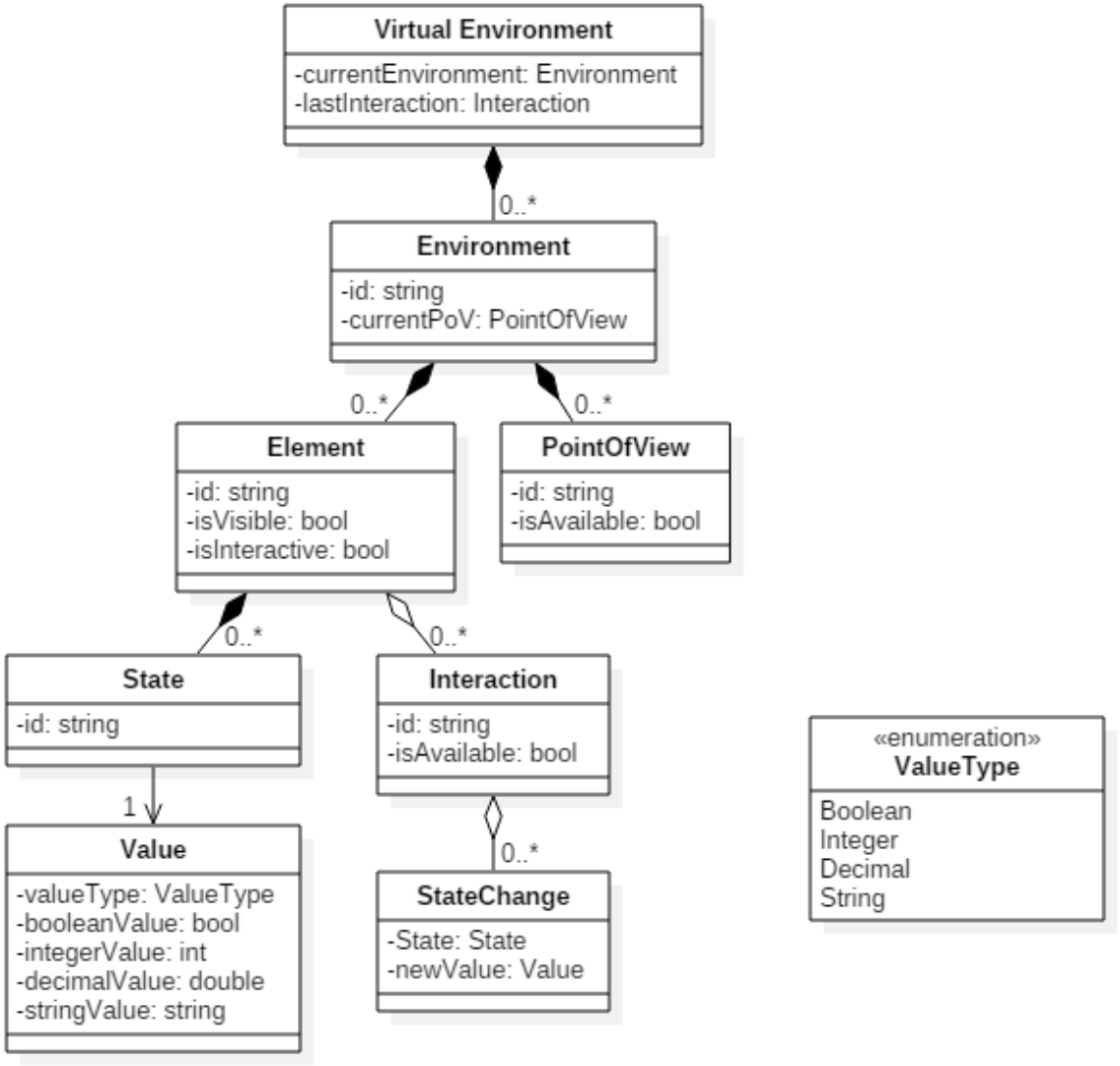


FIGURE 7. MODELE DE REPRESENTATION DE L'ENVIRONNEMENT VIRTUEL

L'état global de l'Environnement Virtuel tel qu'il est représenté dans notre Environnement de Formation est alors défini en partie par la Valeur associée à chaque État de Chaque Objet pour tous les Environnements. Plusieurs propriétés viennent ensuite compléter l'état du système afin que ce même EV puisse être réutilisé dans plusieurs situations d'apprentissage particulières. Ainsi, chaque Objet peut être caché ou affiché, et être rendu interactif (permettant alors d'interdire ou d'autoriser le joueur à utiliser l'Objet associé). Dans le même but, mais de manière plus spécifique, chaque Interaction de chaque Objet peut être rendue disponible ou non. Enfin, il est possible de définir si oui ou non chaque Point de Vue de chaque Environnement est disponible, pour

éventuellement empêcher l'apprenant d'accéder directement à un Objet particulier de l'Environnement associé. Finalement, l'état global de l'EV est complété par la dernière Interaction effectuée par l'utilisateur ainsi que l'Environnement courant et le Point de Vue courant de chaque Environnement.

3.2.2.2 Les Logiciels Métier

Le deuxième module de l'Environnement de Formation directement accessible par l'utilisateur regroupe l'ensemble des Logiciels Métier nécessaires à l'accomplissement des activités étudiées. Le panel des logiciels pouvant être utilisés dans un tel contexte est vaste ; ils peuvent en effet concerner par exemple les logiciels de monitoring médicaux, les logiciels de diagnostic automobiles, ou encore les logiciels de pilotage de machines-outils à commande numérique. On observe alors que ces logiciels ont des domaines d'application ainsi que des objectifs d'utilisation très variés. De plus, à la différence de l'Environnement Virtuel qui a pu être conçu à partir de zéro dans le cadre de nos travaux, les Logiciels Métier que nous ciblons sont des outils déjà existants, qui ont initialement été conçus de manière hétérogène en fonction de leurs besoins respectifs. De ce fait, leur représentation interne diffère, et il n'est pas envisageable pour nous d'imposer aux développeurs de chaque logiciel d'implémenter un modèle spécifique qui corresponde à nos attentes.

Nous avons cependant toujours besoin d'une représentation sur laquelle nous appuyer afin d'orchestrer les activités qui nécessitent l'utilisation de ces LM. Pour cela, l'Environnement de Formation doit être en possession d'un certain nombre d'informations provenant du ou des Logiciels Métiers concernés. Ces informations seront logiquement différentes en fonction du logiciel ciblé (conception et domaine métier différents), et des activités étudiées. Par exemple, un logiciel de monitoring médical pourra vouloir transmettre un certain nombre d'informations en rapport avec les actions qu'il est possible d'effectuer sur un graphique associé à la respiration du patient (définition de courbes référence, comparaison de ces courbes avec la courbe effective, etc.). De son côté, un logiciel de diagnostic automobile devra transmettre des informations de navigation dans le logiciel (écran courant, cliques bouton, …) pour suivre la progression de l'utilisateur dans l'opération courante. Nous avons alors défini un modèle, très général, nous permettant de regrouper l'ensemble de ces informations. Naturellement, ce modèle n'est pas aussi précis que celui de l'Environnement Virtuel, dû justement à l'hétérogénéité des logiciels qui

peuvent être impliqués dans les sessions de formation qui nous intéressent. La Figure 8 présente ce modèle de représentation du module des Logiciels Métiers.

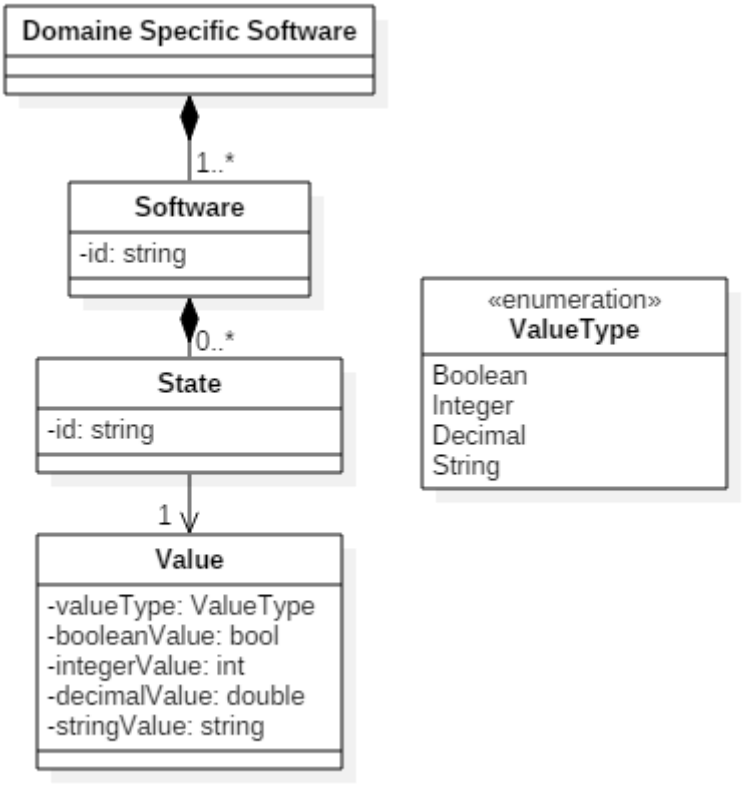


FIGURE 8. MODELE DE REPRESENTATION DES LOGICIELS METIERS

Ainsi, le module est d'abord défini par un ou plusieurs Logiciels. Comme évoqué précédemment, chaque Logiciel est représenté par un ensemble d'informations. Dans notre Environnement de Formation, chacune de ces informations est associée à un ou plusieurs États particuliers qui possèdent chacun une Valeur précise. En reprenant notre exemple du logiciel de monitoring médical, un État intéressant pourrait être la position (x, y) du pointeur de la souris lorsque l'utilisateur est en train de définir une courbe référence. Pour le logiciel de diagnostic, les États intéressants pourraient être l'identifiant de l'écran courant et l'identifiant de la dernière action utilisateur effectuée. Au final, l'état global du module des Logiciels Métier est défini dans l'EF par l'ensemble des États de chaque Logiciel.

3.2.2.3 Les Systèmes de Simulation

Enfin, comme nous l'avons introduit en 3.2.1, le troisième et dernier module qui compose notre Environnement de Formation concerne les systèmes de simulation. En effet, dans certains cas, les Logiciels Métier ont besoin d'échanger des informations avec le monde réel afin de pouvoir fonctionner correctement et/ou de présenter un ensemble de données adaptées au contexte courant. Par exemple, dans la procédure médicale consistant à mesurer les performances respiratoires d'un patient, le logiciel de monitoring doit venir récupérer un certain nombre d'indicateurs sur ce dernier afin de créer les courbes correspondantes et finalement proposer un bilan adéquat. Dans le cas du diagnostic d'une voiture, le garagiste doit utiliser un logiciel métier

qui va venir interroger les différents composants électroniques du véhicule afin d'être en mesure de détecter des disfonctionnements. Dans le cadre de sessions d'apprentissage avec un outil de formation entièrement informatisé, ce besoin d'interaction avec le monde réel devient alors une contrainte forte. En effet, cela implique que la formation ne peut être effectuée par les élèves que s'ils disposent du matériel et/ou du personnel requis, qui en plus doit remplir un certain nombre de conditions pour correspondre à la mise en situation de la leçon (ex : performances respiratoires faibles, panne du moteur, …).

Afin de nous affranchir de ces contraintes, notre Environnement de Formation pourra donc intégrer des Systèmes de Simulations (SS), dont l'objectif est de simuler les différents échanges qui sont censés avoir lieu entre les Logiciels Métier et l'environnement extérieur. Toujours en se basant sur les mêmes exemples, les échanges à reproduire pourraient concerner les valeurs de différents indicateurs pour la mesure de la respiration du patient au cours du temps, ou encore une trame de réponse envoyée depuis un composant électronique du véhicule au logiciel de diagnostic suite à une question particulière sur l'état de ce composant. De la même manière que pour les Logiciels Métier, la conception initiale de ces systèmes et leur représentation interne est très hétérogène afin de répondre à des contraintes et des besoins du domaine métier concerné différents. Ce module de l'Environnement de Formation aura donc aussi un modèle de représentation instancié en fonction des besoins des activités étudiées. La Figure 9 présente ce modèle.

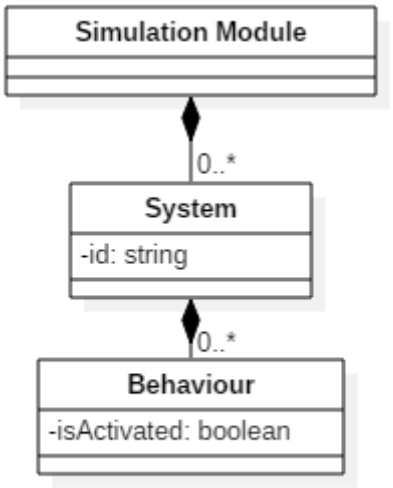


FIGURE 9. MODELE DE REPRESENTATION DES SYSTEMES DE SIMULATION

Ce module peut ainsi être composé de plusieurs Systèmes de simulation (notons qu'il est tout à fait possible que l'EF n'en comporte aucun). Souvent, un Système donné représentera l'environnement responsable de simuler les échanges avec le Logiciel Métier correspondant. Aussi, comme nous l'avons évoqué ci-dessus, l'implémentation effective de ces Systèmes répond à des besoins particuliers, et les échanges avec les Logiciels Métier associés s'inscrivent donc dans un format bien spécifique qu'il ne nous est pas possible d'altérer. Cependant, afin de garder un modèle simple et compréhensible, nous avons souhaité nous abstraire de ce format. En effet, pendant l'orchestration de l'Environnement de Formation, l'objectif ne sera pas de préciser à chaque instant quels échanges doivent être effectués, sous peine de complexifier le processus. Une manière plus simple de décrire l'évolution de ce module serait plutôt de préciser quel

Comportement le système devrait adopter dans certaines situations particulières de la leçon. Dans cette optique, chaque Système peut posséder plusieurs Comportements prédéfinis, chacun pouvant être activé ou désactivé en fonction des actions de l'apprenant tout au long de la session. L'état global de ce module est donc défini par l'état d'activation de chaque Comportement de tous les Systèmes de simulation. A l'exécution de l'Environnement de Formation, chaque Comportement est associé en interne avec un ou plusieurs échanges particuliers qui seront donc effectifs lorsque le Comportement en question sera activé. Cette association, même si elle n'est pas connue du modèle ci-dessus, devra donc être une entrée de notre EF.

3.2.2.4 Interfacer les différents modules

Finalement, notre Environnement de Formation est composé des trois différents modules présentés précédemment. Cependant, chacun de ces modules n'ayant pas été initialement conçus pour fonctionner de concert (en particulier pour l'Environnement Virtuel et les Logiciels Métier), ils ne se synchronisent pas nécessairement entre eux afin de garder un état global et cohérent du système à tout instant. Afin de réponde à cette problématique et d'être en mesure d'interagir uniformément avec chacun des modules par la suite, nous introduisons un élément supplémentaire dans notre Environnement de Formation : le Contexte Partagé (CP), tel que présenté sur la Figure 10.

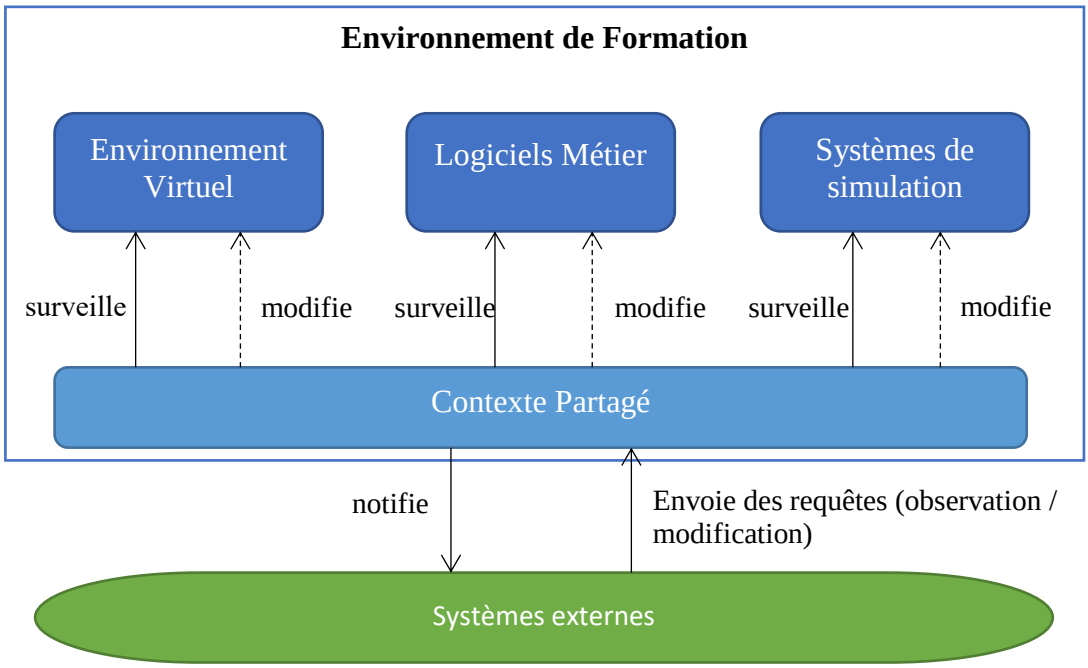


FIGURE 10. ILLUSTRATION DU FONCTIONNEMENT DU CONTEXTE PARTAGE

Le Contexte Partagé fait alors le lien les modèles de représentation des trois modules présents dans l'EF. Le premier rôle du CP, indispensable, est de maintenir un état cohérent du système à tout moment en monitorant l'évolution de chacun de ces modules. Ces derniers devront donc obligatoirement offrir la possibilité de s'y interfacer d'une manière ou d'une autre via un environnement extérieur. Pour cela, plusieurs solutions pourront s'offrir, avec par exemple l'implémentation d'une architecture orientée plug-ins pour l'environnement concerné (Li, Grossman, & Fitzmaurice, 2012) (Grabler, Agrawala, Li, Dontcheva, & Igarashi, 2009), ou encore l'implémentation d'une API permettant l'interopérabilité avec des applications externes, comme

nous le verrons dans le 0. Dans un second temps, le Contexte Partagé sera en mesure, si possible, de modifier manuellement l'état de chaque module. Cela pourra par exemple être utile si l'on veut que la leçon commence dans un état particulier afin d'éviter certaines étapes préliminaires.

Ce Contexte Partagé servira ensuite d'interface avec les environnements extérieurs qui souhaitent interagir avec l'Environnement de Formation. De cette manière, ces environnements seront capables d'utiliser les informations apportées par chaque module de manière uniforme. En particulier, la description des scénarios ainsi que l'outil permettant de les exécuter s'appuiera sur ce CP afin d'orchestrer l'EF tout au long de la leçon comme nous le verrons dans la partie 3.3.

3.2.3 Un outil de formation divertissant

Bien que notre étude se fût initialement portée vers l'utilisation des Jeux Sérieux pour former des élèves à des activités plus ou moins complexes, nous nous sommes aperçus qu'ils ne répondaient pas entièrement aux contraintes posées par le contexte industriel et scientifique qui nous intéressaient. En effet, les activités vers lesquelles nous avons dirigé nos travaux comportent des opérations à effectuer au travers de logiciels, et nous avons alors estimé judicieux d'utiliser directement ces logiciels dans notre Environnement de Formation plutôt que de les simuler dans un environnement virtuel. C'est pourquoi nous avons ensuite élargi notre état de l'art à l'analyse des différentes formes de tutoriels de logiciel, et en particulier les systèmes gamifiés. De ces deux formes d'outil de formations, nous avons alors dégagé un ensemble de particularités. Dans cette partie, nous nous attachons donc à analyser ces caractéristiques afin d'être en mesure de décrire un outil de formation homogène du point de vue pédagogique (3.2.3.1) comme du point de vue ludique (3.2.3.2).

3.2.3.1 La dimension pédagogique

Du point de vue pédagogique, nous avons extrait des Jeux Sérieux et des tutoriels de logiciels plusieurs caractéristiques qui permettent de fournir un contenu de formation pertinent et cohérent. Notre Environnement de Formation étant par définition une combinaison de ces deux formes d'outil d'apprentissage, l'objectif ici est d'analyser leurs attributs respectifs afin d'établir une forme de pédagogie commune et adaptée.

Au niveau des éléments concrets à prendre en compte, le principe d'objectif pédagogique constitue une notion très importante. Dans le cadre des Jeux Sérieux utilisés dans de nombreux domaines et pour de multiples raisons, elle permet d'en catégoriser le contenu pédagogique. En ce qui concerne les tutoriels de logiciel, et plus généralement les activités ciblées par notre Environnement de Formation, cette notion reste toute aussi pertinente même si les objectifs pédagogiques seront le plus souvent liés à des compétences de savoir-faire et cognitives. Un deuxième élément de pédagogie utilisé par les Jeux Sérieux concerne les feedbacks, c'est-à-dire les retours effectués à l'utilisateur à des moments-clé de la leçon pour par exemple insister sur une erreur et tout simplement pour apporter des informations additionnelles vis-à-vis des opérations effectuées. La notion de feedback est aussi présente dans le cadre des tutoriels de logiciel, et est donc directement adaptable dans notre Environnement de Formation. Il sera alors intéressant de proposer différents types de retours pédagogiques, tel que des messages d'aide, d'erreur, ou de conseil (voir Figure 11) Enfin, le principe de résumé de leçon, nécessaire dans tous Jeux Sérieux, est absent des tutoriels de logiciel. Il nous semble cependant pertinent d'intégrer cet élément à notre outil, afin d'être en mesure de revenir sur les points importants de

la leçon et d'effectuer une synthèse globale de la session, et aussi de fournir une évaluation à l'apprenant vis-à-vis de ses performances. Ce résumé pourra être généré automatiquement, en récapitulant à l'apprenant ses erreurs et en lui associant un score en fonction du nombre de mauvaises actions effectuées et du temps qu'il lui a fallu pour terminer la leçon.

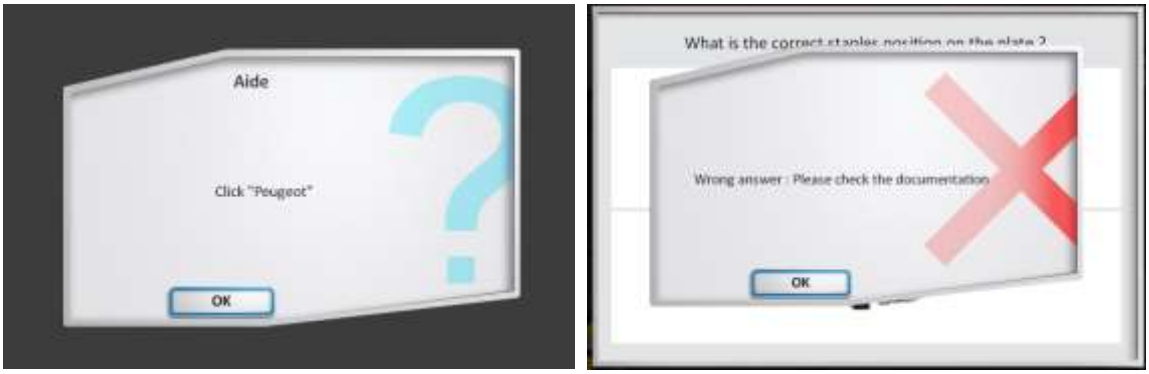


FIGURE 11. EXEMPLES DE RETOURS PEDAGOGIQUES : MESSAGE D'AIDE ET MESSAGE D'ERREUR

Dans un second temps, nous nous intéressons aux règles générales de bonne conception de ces deux outils de formation. Nous avions dégagé des Jeux Sérieux un ensemble de propriétés selon lesquelles l'idée principale était d'impliquer le joueur au maximum dans sa formation pour obtenir des effets positifs sur l'apprentissage. Ce principe est tout aussi pertinent dans les tutoriels de logiciel dont l'objectif est aussi d'enseigner un certain nombre d'opérations clés dans un logiciel spécifique. Ainsi, notre Environnement de Formation devra avoir une faculté de susciter la réflexion chez le joueur, en ne lui indiquant pas systématiquement la marche à suivre pour progresser dans l'activité étudiée. De plus, il permettra à l'apprenant d'explorer son environnement de travail afin d'être en mesure d'analyser les différentes possibilités offertes, ainsi que d'effectuer par lui-même les interactions permettant d'atteindre les objectifs de la leçon et/ou celles qui constituent des erreurs ou des actions inutiles. Dans le même objectif, les systèmes de tutoriel recommandent de concevoir les leçons autour de procédures réelles. Notre outil de formation prendra automatiquement en compte cette propriété puisque les opérations étudiées dans le cadre de ces travaux concernent directement des activités réelles. Un autre grand principe pédagogique mis en place par les Jeux Sérieux et pertinent à intégrer dans notre outil consiste à prendre des précautions pour ne pas perdre l'attention du joueur. Dans cette optique, la difficulté des objectifs pédagogiques devra être progressive et adaptée aux compétences de l'apprenant. De plus, il faudra prendre soin de ne pas intégrer d'éléments contre-productifs tels que des contraintes temporelles ou des éléments pédagogiques qui ne sont pas en rapport direct avec la leçon.

3.2.3.2 La dimension ludique

Du point de vue ludique, nous avons observé au travers de plusieurs exemples que les caractéristiques ludiques des Jeux Sérieux pouvaient servir de base afin de concevoir des tutoriels de logiciel gamifiés. Cependant, alors que les Jeux Sérieux sont, par définition, conçus pour intégrer directement des éléments de jeu, les logiciels qui font l'objet de tutoriels n'ont de leur côté pas été initialement développés avec cet objectif en tête. Gamifier un tel système n'est donc pas toujours une évidence et nécessite une analyse sérieuse afin de pouvoir former un ensemble ludique cohérent.

Le premier point à étudier est l'intégration d'un environnement virtuel en parallèle d'un logiciel afin d'offrir une expérience immersive et engageante à l'apprenant. Cela a été réalisé par exemple dans la suite Office de Microsoft avec Ribbon Hero[25]. Comme nous l'avons vu, cet environnement reproduit de manière cartoon différentes époques, où l'apprenant pourra choisir des missions à la difficulté croissante. Cependant, une fois la mission sélectionnée, cet environnement disparait au profit de l'environnement de travail classique défini par le logiciel requis pour l'accomplissement de cette mission, mais au détriment de la dimension ludique qui s'atténue grandement jusqu'à la sélection de la prochaine mission. D'un autre côté, d'autres systèmes tentent d'intégrer cet environnement virtuel directement dans le logiciel, de manière plus originale. C'est le cas par exemple avec GamiCAD (Li, Grossman, & Fitzmaurice, 2012), où l'environnement virtuel est finalement créé par l'apprenant à travers la création d'objets participant au plaisir ludique (fusée, étoile, etc.). Ce principe n'est cependant disponible que pour les logiciels autorisant une certaine liberté de créativité.

Pour en revenir à notre contexte, les activités qui nous intéressent impliquent déjà un Environnement Virtuel pour simuler des opérations du monde réel, dans lequel il serait possible d'apporter un ensemble d'éléments pour mettre en valeur les dimensions immersives et engageantes de notre outil. Cependant, il nous faut nous assurer que le caractère ludique reste constant à tout instant de la formation, afin de ne pas démotiver l'élève lorsqu'il doit effectuer des opérations sur les logiciels réels, qui eux ne se prêtent pas nécessairement à introduire des éléments qui sortent de leur cadre d'utilisation industriel. Pour cette raison, le caractère immersif et motivant de notre outil ne sera pas déterminé par le caractère de fantaisie identifié dans notre état de l'art. Au contraire, il se fera ressentir en partie par sa nature même, qui permet à l'utilisateur de s'identifier intrinsèquement à son Environnement de Formation, étant une reproduction au plus proche de son cadre de travail habituel. De plus, les mécaniques de jeu implémentées par notre EF, qui participent au plaisir ludique, devront être cohérentes avec la manipulation du logiciel. Dans cette optique, elles s'appuieront sur les jeux d'aventure de type *Point'&'Click*[26], où les interactions entre le joueur et l'environnement de jeu se font en grande majorité avec la souris. Ce type de jeu offre aussi au joueur une certaine liberté, dans le sens où il est libre de se déplacer dans le monde dans le but de trouver le moyen de progresser dans sa mission. Ce principe de liberté d'action, tout à fait adaptable dans l'utilisation d'un logiciel, participera aussi à mettre en valeur l'aspect ludique des formations associées.

Le deuxième point à étudier concerne la liste des différents concepts vidéoludiques que nous avons dressé dans la partie 1.2.3.2. Ces concepts ont la particularité de ne pas être spécifiquement liés à l'environnement virtuel et peuvent donc s'appliquer de manière plus générale dans notre Environnement de Formation. Dans les exemples de tutoriels de logiciel gamifiés que nous avons pu analyser, un de ces éléments ressort presque inévitablement : le principe de missions à accomplir. Bien que chaque mission représente en fait une leçon spécifique, cette simple appellation permet d'introduire plusieurs éléments issus du jeu vidéo qui peuvent s'appliquer dans notre Environnement de Formation dans sa globalité. Ainsi, notre EF devra intégrer un découpage des missions en introduisant un objectif principal et les sous-objectifs associés. Pour renforcer cet aspect, la présentation de ces missions pourra elle-aussi se faire de manière vidéoludique, sous la forme par exemple d'un journal de mission ou d'un journal de quête, comme montré sur la

[25] https://www.microsoft.com/en-us/download/details.aspx?id=26531

[26] Pointes et Cliques en français.

Figure 12. Une autre particularité intéressante associée aux missions dans les jeux vidéo que nous avons relevée est l'augmentation progressive de leur difficulté. Si le domaine métier le permet, l'Environnement de formation devra donc intégrer cette notion. Toujours afin de servir la dimension ludique de notre outil, ces missions pourront être scénarisées et introduites à l'aide de divers moyens comme des vidéos ou des dialogues (Figure 13). Enfin, un score devra être associé à chaque mission en fonction des succès et des erreurs de l'apprenant. Ce score pourra aussi être utilisé en tant que condition pour débloquer les missions suivantes, suscitant une fois de plus des sensations proches de celles que l'on peut retrouver dans les jeux vidéo.

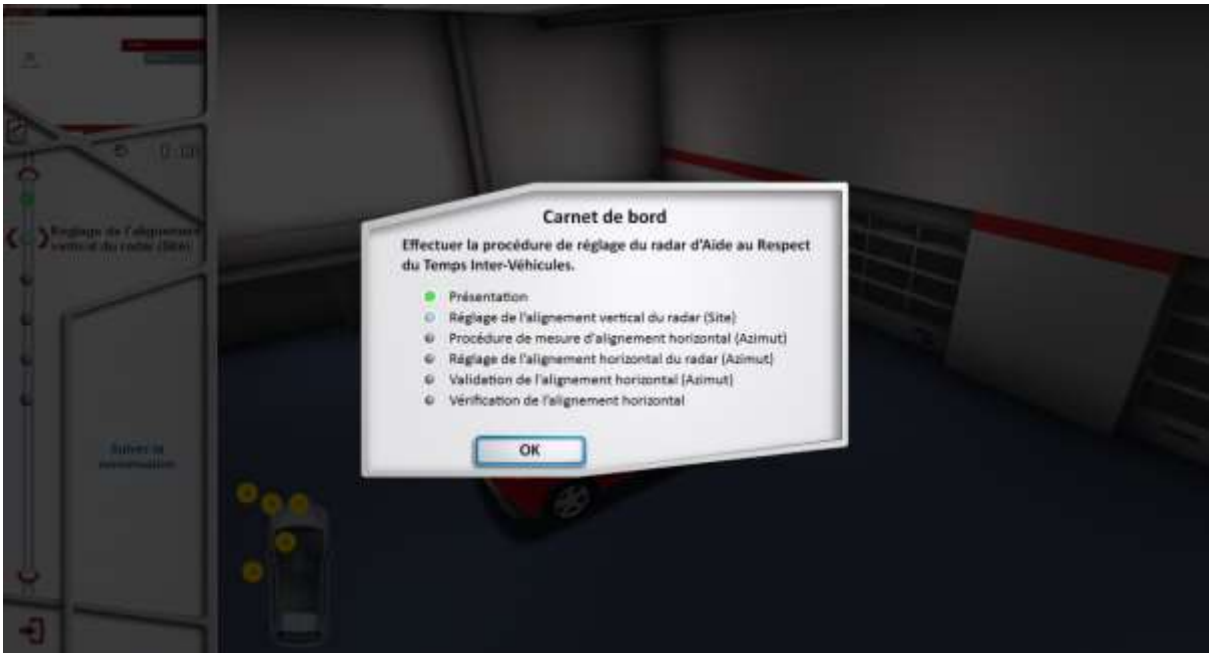


FIGURE 12. EXEMPLE DE REPRESENTATION DES MISSIONS DANS UN CARNET DE BORD

FIGURE 13. EXEMPLE DE MISE EN SCENE DES MISSIONS

3.3 Enseignement d'activités métier dans l'Environnement de Formation : le Scénario de Formation

Maintenant que nous avons défini la nature et les différentes caractéristiques de notre outil de formation, nous nous intéressons dans cette partie aux leçons que devront effectuer les apprenants en évoluant dans l'Environnement de Formation. Après avoir présenté les différentes familles d'activités métier en 3.3.1 et un exemple-type de leçon qui nous intéresse dans la partie 3.3.2 avec ses différentes composantes, nous introduirons alors en 3.3.3.1 la notion de *scénario*, qui correspond justement à la représentation formelle d'une leçon se déroulant dans l'EF. De manière générale, nous présenterons dans la partie 3.3.3 le langage dédié que nous avons défini dans le but de décrire ces scénarios.

3.3.1 Introduction : deux grandes familles d'activités métier à enseigner

Lorsque nous avons conçu notre outil de formation, nous avions initialement imaginé deux grandes familles d'activité métier que ce dernier permettrait d'enseigner :

- les activités **guidées**. Similairement aux tutoriels de logiciel que nous avons étudié dans la partie 1.3, il s'agirait ici pour notre outil de décrire unitairement chaque étape correspondant à la procédure étudiée. La leçon associée serait donc essentiellement linéaire, avec éventuellement à certains moment la possibilité pour l'apprenant de choisir quelle action il souhaite effectuer parmi une liste bien établie. L'important ici est que, par le biais d'une description exhaustive à tout instant des actions disponibles pour l'apprenant, ce type de leçon permettrait d'apporter des retours pédagogiques précis et pertinents en fonction de l'étape courante de la procédure. Cela se ferait cependant au détriment d'une liberté d'action amoindrie, et donc d'une dimension ludique moins présente.
- les activités **libres**. Plus en lien avec la forme des Jeux Sérieux étudiés dans la partie 1.2, il s'agirait ici pour notre outil de laisser l'apprenant libre de toute action afin d'accomplir les objectifs de la leçon enseignée (renforçant ainsi sa dimension ludique). Cette leçon pourrait donc être associée à des procédures métier spécifiques (comme pour les activités guidées), mais aussi à des activités plus générales dont l'objectif pour l'apprenant serait de rechercher les indices nécessaires pour répondre aux problématiques posées initialement. Par exemple, dans le cas d'un client amenant sa voiture au garage à cause d'une panne des feux de croisement, l'apprenant devrait effectuer l'ensemble des procédures de diagnostic nécessaires avant de conclure de la provenance de la panne (ampoule des feux, commande au volant, ou calculateur ?) et de la réparer. Ce type de leçon pourrait alors être décrite par une suite d'étapes optimales (comme pour les activités guidées), mais il ne serait plus possible d'apporter à l'apprenant des retours pédagogiques aussi précis pour lui indiquer l'action à effectuer pour progresser, du fait justement qu'il soit totalement libre. En revanche, il serait possible de lui donner des indices généraux afin de le mettre sur la bonne voie, et de lui fournir à la fin de la leçon un résumé expliquant ses erreurs et comment il aurait pu « mieux faire ».

Avec ces deux familles d'activité en tête, il est possible d'en imaginer une troisième venant mixer les concepts associés à chacune d'elles. En effet, il pourrait être possible de décrire des leçons impliquant à la fois des procédures entièrement guidées et des phases libres où l'apprenant

doit explorer son environnement afin de progresser. Cependant, chacune de ces familles impliquant ses propres problématiques et donc des réflexions différentes, nous avons choisi de nous concentrer sur l'étude de leçons associées à seulement une de ces familles, les activités **guidées**. Ce choix se justifie principalement par le contexte industriel dans lequel s'est déroulé notre étude (voir 0). Dans la suite de ce mémoire, l'ensemble des concepts et des méthodes que nous introduisons sont donc liés à la description de leçons associées à des activités **guidées.**

3.3.2 Leçon : cas d'étude

Afin d'introduire notre méthode de modélisation d'une leçon se déroulant dans l'Environnement de Formation présenté en 3.2, nous en présentons dans la Figure 14 un exemple-type. Cela nous servira par la suite à isoler les composantes qu'il nous faudra décrire dans nos scénarios.

Réglage du radar anticollision sur le véhicule Peugeot 308

I – Déroulement de la leçon (début)

1. Dans n'importe quel ordre :

a) Établir la communication avec le véhicule (à l'aide du logiciel de diagnostic)

i. Sur l'écran de sélection de la marque du véhicule, cliquer sur Peugeot
ii. Sur l'écran de sélection du modèle du véhicule, l'apprenant a le choix entre
   - Choisir le véhicule manuellement
     - Cliquer sur le bouton 308
     - Cliquer sur le bouton OK dans la pop-up qui apparait
   - Détecter le véhicule automatiquement
     - Cliquer sur le bouton de détection automatique
     - Cliquer sur OK sur l'écran suivant une fois que le véhicule a été détecté
iii. Sur l'écran qui apparait ensuite, cliquer sur OK.

b) Préparer le radar anticollision (à l'aide de l'Environnement Virtuel)

i. Démonter le pare-chocs avant
ii. Positionner automatiquement la caméra devant le radar anticollision
iii. Placer la plaque adaptée sur le radar anticollision

II – Pédagogie

- Objectifs pédagogiques
  - L'étape *a) Établir la communication avec le véhicule* est liée à l'objectif pédagogique *Savoir manipuler le logiciel de diagnostic*.
  - L'étape *b) Préparer le radar anticollision* est liée à l'objectif pédagogique *Savoir opérer sur le véhicule*.

- Retours pédagogiques
  - Fournir une aide associée à chaque action devant être réalisée par l'apprenant.
  - Pour chaque erreur effectuée par l'apprenant, afficher le message d'erreur générique suivant : *"Erreur : cette action ne permet pas de progresser dans l'activité en cours."*

III – Éléments ludiques

- Mission : Effectuer le début de la procédure de réglage du radar anticollision de la Peugeot 308 d'un client qui vient d'avoir un accrochage avec un autre véhicule.

- Sous-missions : Les étapes *a) Établir la communication avec le véhicule* et *b) Préparer le radar anticollision* correspondent chacune à une sous-mission.

- Difficulté de la mission : 1 / 3

- Score requis pour valider la leçon : 80% (4 étoiles sur cinq).

- Mises en scène
  - Au début de la leçon : dialogue entre le garagiste et le client
  - A la fin de la leçon : dialogue entre le garagiste et le client

FIGURE 14. EXEMPLE-TYPE DE LEÇON ETUDIEE A TRAVERS NOTRE OUTIL DE FORMATION

La leçon étudiée propose de former un garagiste à la procédure de réglage du radar anticollision sur la Peugeot 308. Ce radar, présent sur les modèles récents de ce véhicule, permet entre-autres de mesurer la distance entre la voiture du conducteur et la voiture qui le précède. Dans certaines circonstances (accident par exemple), il peut arriver que ce radar se désaligne. Si c'est le cas, les fonctionnalités proposées par le radar sont désactivées, et le propriétaire du véhicule doit se déplacer en garage afin qu'un technicien effectue une procédure de réglage visant à réaligner le radar. Cette procédure implique alors d'effectuer des opérations sur le véhicule en suivant les indications données par un logiciel de diagnostic, et rentre donc dans le cadre de nos recherches. Étant une procédure relativement longue, nous ne nous intéressons ici qu'aux opérations préliminaires associées où le garagiste doit effectuer deux grandes activités dans un ordre non-imposé (la leçon complète est donnée en Annexe 1) :

- Établir la communication avec le véhicule à l'aide du logiciel de diagnostic. En situation réelle, une Interface de Communication avec le Véhicule (VCI) doit être utilisée afin de

permettre au logiciel de récupérer des informations des différents systèmes électroniques du véhicule. Dans le cadre de notre Environnement de Formation, un logiciel est utilisé afin de simuler cette VCI (cf. : Systèmes de simulation, 3.2.2.3), et nous supposons que la VCI est déjà installée dans l'Environnement Virtuel.

- Préparer le radar à la procédure d'alignement du radar (non-étudiée dans cet exemple). Pour cela, le garagiste doit démonter le pare-chocs afin d'accéder au radar et placer une place spécifique permettant par la suite de poser un inclinomètre et d'effectuer des réglages corrects.

Sur cet exemple, nous pouvons observer que les leçons qui nous intéressent sont composées de trois grandes parties :

- Description du déroulement de la leçon. Elle précise d'un côté les actions devant être réalisées par l'apprenant dans l'Environnement de Formation afin d'accomplir les différentes activités de la leçon, en proposant éventuellement des opérations alternatives et/ou des opérations pouvant être effectuées sans contrainte d'ordre. Cela permet entre autres d'accentuer l'aspect ludique de l'outil en proposant une certaine notion de liberté et de rejouabilité. D'un autre côté, cette description permet de préciser les modifications devant être effectuées sur l'Environnement de Formation automatiquement par l'outil à un instant précis de la leçon. Cela permet d'animer l'EF et ainsi de placer l'élève dans des conditions particulières, afin par exemple d'insister sur des éléments importants liés à la leçon.
- Description de la pédagogie. Dans un premier temps, elle permet de représenter les objectifs pédagogiques liés aux activités métier étudiées au travers de la leçon. Dans le cas présent, un seul objectif pédagogique est associé aux deux grandes activités de la leçon, mais il serait tout à fait possible d'en faire une description plus précise, aussi bien dans le découpage des objectifs pédagogiques eux-mêmes que dans les opérations associées. Dans un second temps, elle permet de décrire les retours pédagogiques qui devront être apportés à l'apprenant tout au long de la leçon et à des instants particuliers. Ici, des messages d'aides sont précisés à chaque action devant être effectuée par l'apprenant dans l'Environnement de Formation, et un message d'erreur générique est affiché à chaque action effectivement effectuée par l'élève mais ne faisant pas progresser la leçon. Notons qu'aucune information liée au résumé de la leçon à fournir n'est décrite dans cette partie car l'Environnement de Formation cible sera en mesure d'en générer un automatiquement (comme décrit en 3.2.3.1).

Description des éléments ludiques. Il s'agit ici de décrire les différents éléments liés à la dimension ludique de notre environnement, et en particulier à la scénarisation et contextualisation de la leçon par le biais de la notion de mission. L'énoncé général de la mission est alors précisé, ainsi que les différentes sous-activités (= sous-missions / sous-quêtes) qui lui sont liées. De plus, une difficulté et un score minimal permettant de valider la leçon lui sont associés. Enfin, des mises en scènes peuvent être décrites afin de scénariser de manière plus importante la mission, mais aussi afin de reproduire des conditions réalistes vis-à-vis du déroulement des activités de la leçon.

3.3.3 Scénario de Formation

3.3.3.1 Définition

L'exemple de leçon présenté en 3.3.2 nous a permis d'identifier les grandes dimensions qui composent les activités à enseigner qui nous intéressent. L'objectif dans cette partie est alors d'introduire une méthode permettant de décrire formellement une leçon afin que l'apprenant puisse suivre les enseignements associés en évoluant dans notre Environnement de Formation. Nous introduisons alors la notion de S*cénario de Formation* (que nous simplifierons par *scénario* dans la suite de ce manuscrit), que nous définissons comme tel dans le cadre de nos travaux :

> *Un Scénario de Formation est la représentation formelle d'une leçon se déroulant dans notre Environnement de Formation. Elle implique :*
>
> - *la description de l'Activité, c'est-à-dire l'évolution de l'Environnement de Formation en fonction des actions de l'apprenant ;*
> - *la description de la dimension Pédagogique, avec les objectifs et retours pédagogiques associés. Ces éléments pédagogiques sont disponibles dans une base de données associée.*
> - *la description de la dimension Ludique, avec tous les éléments associés à la notion de mission permettant de contextualiser et de scénariser la leçon. Ces éléments ludiques sont disponibles dans une base de données associée.*

Dans les parties qui suivent, nous nous intéressons alors à définir un langage dédié permettant de représenter un scénario. Pour cela, nous nous appuierons sur les différents modèles décris en 3.2 et qui caractérisent notre Environnement de Formation, car ils représentent les seuls éléments formels introduits jusqu'à présent permettant d'établir le lien entre la leçon et l'environnement informatique dans lequel l'apprenant pourra accomplir les différentes activités liées à cette leçon.

3.3.3.2 Description de l'Activité

Afin de décrire l'Activité et tous les éléments qui lui sont associés, nous allons tout d'abord isoler les différentes notions du langage naturel que l'on retrouve dans l'exemple donné dans la partie 3.3.2. Chacune de ces notions permet d'exprimer différentes idées qu'il doit être possible de représenter dans un scénario, afin de donner à notre langage de modélisation l'expressivité nécessaire et suffisante pour décrire les différentes activités de la leçon. Nous donnons ci-dessous la liste de ces notions, avec leur formalisation respective en instruction dans notre langage dédié :

- ***Points de passage***. La première notion que nous souhaitons pouvoir représenter est la description de points de passage spécifiant les instants-clé faisant progresser l'élève dans le scénario. Ces points de passages peuvent alors être utilisés pour spécifier l'action du joueur attendue à un moment précis (ex : le joueur doit cliquer sur le bouton Peugeot dans le logiciel de diagnostic), mais ils peuvent aussi être utilisés afin de décrire une situation plus générale devant être atteinte avant de pouvoir accomplir la suite du scénario (ex : la plaque est correctement positionnée sur le radar et la communication avec le véhicule est établie). Dans le premier cas, c'est l'attribut *lastInteraction* du modèle de l'Environnement Virtuel (Figure 7) ou des Logiciels Métier (Figure 8) qui sera utilisé afin d'identifier l'action attendue. Dans le deuxième cas, nous utiliserons les

différentes propriétés de chacun des modules qui permettent de représenter leur état global dans le cadre de l'Environnement de Formation. La notion de point de passage est représentée avec l'instruction suivante dans notre langage dédié :

```
Checkpoint ([conditions]);
```

- **conditions** est une combinaison de **conditions** utilisant les opérations booléennes classiques (ET, OU, OU EXCLUSIF).
- **condition** est définie sous la forme suivante

```
[propertyIdentifier][booleanComparator][propertyValue]
```

- **propertyIdentifier** est l'identifiant de la propriété dans l'Environnement de Formation à tester (ex : `softs.soft_1.lastInteraction`).
- **booleanComparator** est l'opérateur de comparaison utilisé (`==`, `>=`, `>`, `<=`, `<`, `!=`).
- **propertyValue** est la valeur à laquelle est comparée la propriété identifiée par **propertyIdentifier** (ex : `"PeugeotButtonClicked"`). Chaque propriété étant typée, cette valeur doit être de même type que la propriété testée.

- ***Animation***. Une deuxième notion à représenter concerne la volonté de décrire, à certains moments de la leçon, une animation automatique de l'Environnement de Formation. Cela peut être utile afin de placer le joueur dans une situation de formation particulière (ex : dans le système de simulation des communications entre le véhicule et le logiciel de diagnostic, activer le comportement "radar anticollision désaligné"), ou encore afin de montrer un élément particulier dans l'Environnement de Formation (ex : positionner la caméra devant le radar anticollision). Ici aussi, ce sont les différentes propriétés de chaque module qui seront utilisées afin de décrire ces modifications manuelles de l'état de l'EF. Notre langage dédié représente la notion d'animation avec l'instruction suivante :

```
Animation ([propertyIdentifier] = [newValue]);
```

- **propertyIdentifier** est l'identifiant de la propriété dans l'Environnement de Formation à modifier (ex : `ve`[27]`.currentEnvironment.currentPoV`).
- **newValue** est la nouvelle valeur que doit prendre la propriété identifiée par **propertyIdentifier** (ex : `"anticollisionRadar"`). Chaque propriété étant typée, cette valeur doit être de même type que la propriété testée.

- ***Choix***. La troisième notion qu'il est nécessaire de pouvoir représenter dans la description de nos scénarios est la notion de choix. L'objectif ici est de pouvoir proposer à des instants précis durant le déroulement de la leçon un choix à l'apprenant concernant les activités à effectuer par la suite. Ces choix se déclinent sous deux formes différentes :
  - ***Activités alternatives***. Il s'agit ici de représenter la volonté de proposer au joueur plusieurs activités permettant d'accomplir une même tâche (ex : afin de spécifier le véhicule à diagnostiquer, l'apprenant peut soit le détecter automatiquement, soit le sélectionner manuellement dans le logiciel associé). Les activités alternatives se déclarent avec l'instruction suivante :

[27] ve = Environnement Virtuel (Virtual Environment en anglais)

```
Alternative_Activities
{
    …
    Déclaration des branches
    …
}
```

- ***Activités concurrentes***. Ce type de choix permet de décrire plusieurs activités devant être effectuées avant de pouvoir continuer la leçon, mais sans imposer de contrainte chronologique dans l'accomplissement de ces activités. Le joueur est ainsi laissé libre de choisir l'ordre dans lequel ils souhaitent effectuer les opérations liées à chaque activité. Il est aussi possible de spécifier des activités facultatives ne devant pas obligatoirement être accomplies afin de pouvoir effectuer la suite de la leçon. Les activités concurrentes se déclarent avec l'instruction suivante :

```
Parallel_Activities ([must_be_performed_branches])
{
    …
    Déclaration des branches
    …
}
```

- **`must_be_performed_branches`** est la liste des **`branches`** devant être effectivement accomplies afin de pouvoir effectuer la suite du scénario et se décrit comme cela : `branch_1 && branch_2 && … branch_n`.

Chacune de ces formes impliquent alors la déclaration de ***branches*** afin de décrire les activités correspondant à chaque choix. Chacune des branches doit spécifier une condition sur une ou plusieurs propriétés de l'Environnement de Formation afin de définir celle que l'élève commence à effectuer. Une branche se déclare avec l'instruction suivante dans notre langage dédié :

```
Branch ([branch_id], [conditions])
{
    [instructions]
}
```

- **`branch_id`** est l'identifiant de la branche déclarée (ex : `branch_1`)
- **`conditions`** est une combinaison de **conditions** et se définit de la même manière que pour l'instruction `Checkpoint`.
- **`instructions`** est une suite d'instructions permettant de décrire l'activité associée à la branche. Toutes les instructions présentées dans cette partie sont donc utilisables (sauf l'instruction Branch qui ne peut être utilisée que dans le cadre des activités alternatives ou parallèles).

- ***Fin de scénario***. La dernière notion nécessaire dans la description de l'Activité est celle qui permet de spécifier quand est-ce que l'apprenant a terminé les activités du scénario. Chaque scénario possède nécessairement au moins une instruction de fin de scénario à la toute fin, mais il est aussi possible de définir des fins "alternatives" dans les branches des activités alternatives et concurrentes. Cela peut par exemple être utile pour représenter un "game over" si une des branches déclarées représente en fait un chemin

d'erreur sur lequel on souhaitait insister dans notre scénario. Afin de représenter une fin de scénario, il suffit d'utiliser l'instruction suivante :

```
End ();
```

En utilisant l'ensemble des instructions de notre langage dédié présentées ci-dessus, il nous est maintenant possible de décrire l'exemple présenté dans la partie 3.3.2 :

```
Parallel_Activities(branch_1 && branch_4)
{
    Branch(activity_1, softs.diag.lastInteraction == "PEUGEOT")
    {
        Alternative_Activities
        {
            Branch (branch_2, softs.diag.lastInteraction == "308")
            {
                Checkpoint (softs.diag.lastInteraction == "OK");
            }
            Branch (branch_3, softs.diag.lastInteraction == "AUTO_DETECTION")
            {
                Checkpoint (softs.diag.lastInteraction == "OK");
            }
        }
    }

    Branch (branch_4, ve.lastInteraction == "GARAGE.BUMPER.TAKE_OFF")
    {
        Animation (ve.pov = "RADAR_POV");
        Checkpoint (ve.lastInteraction == "GARAGE.RADAR.PUT_PLATE_ON");
    }
}


End ();
```

Finalement, les instructions décrites dans cette partie sont suffisantes afin de représenter formellement les différentes activités d'une leçon semblable à celle proposée en 3.3.2. On peut alors observer l'utilisation des différentes propriétés de notre Environnement de Formation afin de décrire son évolution. Ici, ce sont principalement les propriétés *lastInteraction* de l'Environnement Virtuel et du Logiciel Métier de diagnostic respectivement qui sont utilisées. Cela est en effet adapté pour la leçon considérée qui a pour but d'enseigner étape par étape les actions nécessaires pour l'accomplissement d'une procédure précise, et permet d'être en mesure de détecter les actions inutiles et/ou erronées pour y réagir à l'exécution du scénario. De plus, on note l'utilisation des deux types d'instruction de choix afin d'offrir à l'élève des alternatives dans le déroulement de la leçon :

- Les activités concurrentes pour les deux grandes activités du scénario qui peuvent s'effectuer dans n'importe quel ordre (établir la communication avec le véhicule et préparer le radar anticollision).
- Les activités alternatives pour la sélection du véhicule à l'aide du logiciel de diagnostic (manuelle ou automatique).

3.3.3.3 Description de la dimension Pédagogique

Dans le cadre des leçons étudiées, la description de la dimension Pédagogique concerne la spécification des objectifs pédagogiques et des retours pédagogiques, ainsi que l'association de ces éléments avec le déroulement de la leçon afin de faire le lien avec la progression de l'apprenant dans le scénario. Pour chacun de ces éléments, nous introduisons alors le format XML qui nous permet de les représenter formellement dans nos scénarios, ainsi que l'instruction de notre langage dédié nous permettant de les lier avec la description de l'Activité :

- ***Objectifs Pédagogiques***. Les objectifs pédagogiques permettent de représenter les différentes connaissances devant être acquises par l'apprenant à la fin des différentes activités proposées par le scénario (ex : *Connaitre les opérations préliminaires de la procédure d'alignement du radar anticollision*). Dans le cadre de nos scénarios, nous avons besoin d'être en mesure de définir autant d'objectifs pédagogique que souhaité ; pour cela, nous utilisons le format XML afin de les spécifier formellement dans la base de données associée :

```
<PedagogicalObjective id = [objectiveId]>
    <Label language = "fr"> [labelText] </ Label>
    …
</PedagogicalObjective>
```

- **`objectiveId`** est l'identifiant de l'objectif qui sera utilisé lors de l'association avec l'Activité.
- **`labelText`** est le libellé de l'objectif qui indique à l'apprenant la connaissance transmise. Notons qu'il est possible de définir plusieurs balises `Label` afin de définir le libellé dans plusieurs langues.

Le lien avec la description de l'Activité est ensuite décrit dans notre langage dédié à l'aide de l'instruction suivante :

```
Pedagogical_Objectives ([objectivesId])
{
    [instructions]
}
```

- **`objectivesId`** est une liste d'identifiant des objectifs associés et se décrit comme cela : `objectiveId_1 && objectiveId_2 && … & objectiveId_n`.
- **`instructions`** est la suite d'instructions (seule l'instruction `Branch` n'est pas utilisable) qui décrit les objectifs pédagogiques associés. Concrètement, cela implique que toutes les opérations effectuées par l'apprenant dans le cadre de ces instructions sont liées à ces objectifs pédagogiques. Lors du résumé de la leçon, le calcul du score lié à ces objectifs prendra donc en compte ces opérations.
- En ce qui concerne l'imbrication d'instructions `Pedagogical_Objective`, cela n'a de sens que si la liste des objectifs associés est strictement une sous-liste de la liste spécifiée par l'instruction parente (même si la grammaire de notre langage permet le contraire). En effet, de manière logique, une suite d'opérations liées à l'objectif 1 et à l'objectif 2 ne peut pas être décomposée pour être associée à un objectif 3 ; de plus, la décomposition de ces opérations n'est utile que si l'objectif 1 et l'objectif 2 sont spécifiés chacun de leur côté.

- ***Retours Pédagogiques***. Les retours pédagogiques représentent les différentes informations qui seront apportées à l'apprenant tout au long du scénario (ex : action inutile dans le cadre de l'activité en cours, message d'assistance pour aider l'élève à progresser dans la leçon). De la même manière que pour les objectifs pédagogiques, autant de retours pédagogiques que nécessaires pourront être définis, en utilisant le format XML suivant :

```
<PedagogicalFeedback type = [typeId] id = [feedbackId]>
    <Property id = [propertyId] value = [value]>
        <TranslatableValue language = "fr"> [translatableValue]
        </ TranslatableValue>
        …

        <Property …>
            …
        </Property>
        …
    </ Property>
    …
</ PedagogicalFeedback >
```

- **typeId** est le type du retour pédagogique (ex: erreur, aide, …).
- **feedbackId** est l'identifiant du retour pédagogique qui sera utilisé lors de l'association avec l'Activité.
- Chaque feedback est défini par un certain nombre de propriétés (balise `Property`). A chacune de ces propriétés est associé un identifiant (**propertyId**) et une valeur facultative (**value**). De plus, une propriété peut éventuellement représenter un texte traductible ; dans ce cas, une balise `TranslatableValue` est utilisée pour chacun des traductions (**translatableValue**). De plus, chaque propriété peut elle-même contenir des sous-propriétés, et ainsi de suite.

Le lien avec la description de l'Activité est ensuite décrit dans notre langage dédié à l'aide de l'instruction suivante :

```
PedagogicalFeedback (feedbackId);
```

- **feedbackId** est l'identifiant du retour pédagogique à exécuter.

Finalement, en reprenant la description de l'Activité donnée en 3.3.3.2, l'exemple de la partie 3.3.2 se représente formellement de la manière suivante (les différents éléments pédagogiques utilisés ici sont décrits en Annexe 2) :

```
PedagogicalFeedback(global_error);
PedagogicalFeedback(help_1);
Parallel_Activities(branch_1 && branch_4)
{
    Branch(branch_1, softs.diag.lastInteraction == "PEUGEOT")
    {
        Pedagogical_Objectives(objective_1)
        {
            PedagogicalFeedback(help_2);
            Alternative_Activities
            {
                Branch (branch_2, softs.diag.lastInteraction == "308")
                {
                    PedagogicalFeedback(help_3);
                    Checkpoint (softs.diag.lastInteraction == "OK");
                }
                Branch (branch_3, softs.diag.lastInteraction ==
"AUTO_DETECTION")
                {
                    PedagogicalFeedback(help_4);
                    Checkpoint (softs.diag.lastInteraction == "OK");
                }
            }
        }
    }

    Branch (branch_4, ve.lastInteraction == "GARAGE.BUMPER.TAKE_OFF")
    {
        Pedagogical_Objectives(objective_2)
        {
            Animation (ve.pov = "RADAR_POV");
            PedagogicalFeedback(help_5);
            Checkpoint (ve.lastInteraction == "GARAGE.RADAR.PUT_PLATE_ON");
        }
    }
}

End ();
```

On note alors l'utilisation des instructions `PedagogicalObjectives` et `PedagogicalFeedback` directement dans la description de l'Activité, car ils sont intimement liés à la progression de l'apprenant dans la leçon. De plus, chaque type de retour pédagogique (ici : message d'accompagnement et message d'aide) est traité spécifiquement par le moteur qui exécute le scénario, et leur utilisation dans la description du scénario est donc dépendante de ce traitement. Ici, on suppose que le message d'erreur assigné sera exécuté à chaque prochaine action erronée ou inutile de l'apprenant. Dans le cadre de cet exemple, étant donné que le message d'erreur reste le même tout du long de la leçon, nous n'avons alors besoin de l'assigner qu'au tout début du scénario. En ce qui concerne les messages d'aide, ils représentent un texte auquel l'utilisateur aura accès en cliquant sur un bouton. Pour cette raison, nous assignons le message d'aide avant chaque instruction représentant une action devant être effectuée par le joueur.

3.3.3.4 Description de la dimension Ludique

Enfin, la description de la dimension Ludique concerne la spécification de plusieurs éléments qui ont un impact sur le déroulement de la leçon (sous-missions, mise en scène), ou non (description de la mission, niveau de difficulté, score requis pour valider la leçon). Pour les premiers, nous introduisons, à l'instar que pour les éléments pédagogiques, le format XML nous permettant de les décrire formellement dans le cadre de nos scénarios, avec l'instruction de notre langage dédié permettant de décrire le lien avec la description de l'Activité. Pour les deuxièmes, nous ne présentons que le format XML correspondant.

- ***Mission, niveau de difficulté et score de validation***. Il s'agit ici de représenter de manière ludique les éléments se rapportant au principe de mission n'ayant pas de lien avec le déroulement du scénario en lui-même. Pour cela le format XML suivant est utilisé afin de les spécifier dans la base de données associée :

```
<Mission difficulty = [level] validatingScore = [score] >
    <Label language = "fr"> [labelText] </ Label>
    …
</Mission>
```

- **`level`** représente le niveau de difficulté de la mission.
- **`validatingScore`** est le score requis pour valider la mission et éventuellement débloquer les missions suivantes.
- **`LabelText`** est l'énoncé de la mission, traduisible en plusieurs langues à l'aide de plusieurs balises Label.

- ***Sous-missions***. La mission peut ensuite être décomposée en sous-missions, qui correspondent concrètement à plusieurs activités que le joueur devra effectuer afin de terminer la leçon. Chacune des sous-missions doit alors être définie dans un premier temps en utilisant le format XML suivant :

```
<Sub_Mission id = [sub_mission_id]>
    <Label language = "fr"> [labelText] </ Label>
    …

    <Sub_Mission …>
        …
    </Sub_Mission>
    …
</Sub_Mission >
```

- **`sub_mission_id`** est l'identifiant de la sous-mission considérée.
- **`LabelText`** est l'énoncé de la sous-mission, traduisible en plusieurs langues à l'aide de plusieurs balises `Label`.
- Il est aussi possible de définir des sous-missions à la sous-mission courante en utilisant la balise `Sub_Mission`, et ainsi de suite.

Chacune des sous-missions doit ensuite être associée à la description de l'Activité afin de les valider au fur et à mesure de la progression du joueur dans les opérations de la leçon. Cela se fait en utilisant l'instruction suivante de notre langage dédié :

```
Sub_Mission ([sub_mission_id])
{
    [instructions]
}
```

- **sub_mission_id** est l'identifiant de la sous-mission considérée.
- **instructions** est la suite d'instructions (seule l'instruction `Branch` n'est pas utilisable) qui décrit la sous-mission associée. Concrètement, cela implique que toutes les opérations effectuées par l'apprenant dans le cadre de ces instructions sont liées à cette sous-mission. Lors du résumé de la leçon, le calcul du score lié à cette sous-mission prendra donc en compte ces opérations.
- En ce qui concerne l'imbrication d'instructions `Sub_Mission`, cela n'a de sens que si la mission associée est strictement une sous-mission de la mission spécifiée par l'instruction parente (même si la grammaire de notre langage permet le contraire).

- ***Mise en scène***. Enfin, la mission peut être scénarisée en utilisant différents éléments de mise en scène (ex : dialogues, vidéos, ...). Chacun de ces éléments doit d'abord être spécifié en utilisant le format XML suivant :

```
<Staging type = [typeId] id = [stagingId]>
    <Property id = [propertyId] value = [value]>
        <TranslatableValue language = "fr"> [translatableValue]
        </TranslatableValue>
        …

        <Property …>
            …
        </Property>
        …
    </Property>
    …
</Staging>
```

- **typeId** est le type de mise en scène (ex: dialogue, vidéo, …).
- **stagingId** est l'identifiant de la mise en scène qui sera utilisée lors de l'association avec l'Activité.
- Chaque élément de mise en scène est défini par un certain nombre de propriétés (balise `Property`). A chacune de ces propriétés est associé un identifiant (**propertyId**) et une valeur facultative (**value**). De plus, une propriété peut éventuellement représenter un texte traductible ; dans ce cas, une balise `TranslatableValue` est utilisée pour chacun des traductions (**translatableValue**). De plus, chaque propriété peut elle-même contenir des sous-propriétés, et ainsi de suite.

Afin de définir le moment où chacun de ces éléments de mise en scène sera exécuté durant le scénario, il est nécessaire de les associer à la description de l'Activité. Pour cela, notre langage dédié utilise l'instruction suivante dans la description de notre scénario :

```
Staging (stagingId);
```

- **stagingId** est l'identifiant de l'élément de mise en scène à exécuter.

Finalement, nous pouvons compléter et terminer le scénario qui modélise l'exemple de leçon donné en 3.3.2 (les missions et éléments de mise en scène utilisés ici sont décrits en Annexe 2) :

```
PedagogicalFeedback(global_error);
Staging(dialog_1);
PedagogicalFeedback(help_1);
Parallel_Activities(branch_1 && branch_4)
{
    Branch(branch_1, softs.diag.lastInteraction == "PEUGEOT")
    {
        Pedagogical_Objectives(objective_1)
        {
            Sub_Mission(sub_mission_1)
            {
                PedagogicalFeedback(help_2);
                Alternative_Activities
                {
                    Branch (branch_2, softs.diag.lastInteraction == "308")
                    {
                        PedagogicalFeedback(help_3);
                        Checkpoint (softs.diag.lastInteraction == "OK");
                    }
                    Branch (branch_3, softs.diag.lastInteraction ==
"AUTO_DETECTION")
                    {
                        PedagogicalFeedback(help_4);
                        Checkpoint (softs.diag.lastInteraction == "OK");
                    }
                }
            }
        }
    }

    Branch (branch_4, ve.lastInteraction == "GARAGE.BUMPER.TAKE_OFF")
    {
        Pedagogical_Objectives(objective_2)
        {
            Sub_Mission(sub_mission_2)
            {
                Animation (ve.pov = "RADAR_POV");
                PedagogicalFeedback(help_5);
                Checkpoint(ve.lastInteraction=="GARAGE.RADAR.PUT_PLATE_ON");
            }
        }
    }
}

Staging(dialog_2);


End ();
```

On note alors que la description des éléments ludiques par rapport au déroulement de la leçon se fait uniquement en faisant appel à des instructions "simples" aux moments adéquats par rapport à l'avancée du joueur dans les opérations qu'il doit réaliser durant la leçon.

3.4 Synthèse

Tout au long de ce chapitre, nous avons spécifié les différentes composantes qui forment notre outil de formation. Dans un premier temps, nous nous sommes intéressés à définir l'Environnement de Formation dans lequel l'apprenant sera amené à évoluer afin d'effectuer des activités métier spécifiques. Ces activités métier nécessitant d'effectuer des opérations à la fois sur des objets du monde réel et dans des logiciels particuliers, l'environnement que nous avons conçu est constitué de plusieurs modules :

- L'Environnement Virtuel, qui permet de simuler le monde réel avec ses différents objets et interactions ;
- Les Logiciels Métier, qui sont nécessaires à l'accomplissement des activités métier étudiées ;
- Les Systèmes de simulation, qui permettent de simuler les informations échangées entre le monde réel et les Logiciels Métier associés.

Ces modules sont alors représentés par plusieurs modèles, pour ensuite être interfacés via un Contexte Partagé dans le but d'uniformiser les interactions avec chacun de ces modules. Nous avons ensuite introduit les dimensions pédagogiques et ludiques de notre environnement de formation, en présentant différents éléments et concepts issus des Jeux Sérieux et des tutoriels de logiciel.

Dans un second temps, nous avons introduit la notion de Scénario de Formation :

> *Un Scénario de Formation est la représentation formelle d'une leçon se déroulant dans notre Environnement de Formation. Elle implique :*
>
> - *la description de l'Activité, c'est-à-dire l'évolution de l'Environnement de Formation en fonction des actions de l'apprenant ;*
> - *la description de la dimension Pédagogique, avec les objectifs et retours pédagogiques associés. Ces éléments pédagogiques sont disponibles dans une base de données associée.*
> - *la description de la dimension Ludique, avec tous les éléments associés à la notion de mission permettant de contextualiser et de scénariser la leçon. Ces éléments ludiques sont disponibles dans une base de données associée.*

Au travers d'un cas d'étude, nous avons alors explicité un langage dédié se reposant sur les propriétés de notre Environnement de Formation afin de décrire de manière exhaustive les différentes composantes de leçons associées à des activités guidées.

Chapitre 4. Vers une méthodologie orientée-formateur pour la création de Scénarios de Formation

4.1 Introduction

Le langage dédié que nous avons introduit en 3.3.3 pour décrire des Scénarios de Formation a été conçu afin d'être directement interprétable par des systèmes informatiques, dans la finalité d'enseigner des activités métier au travers de notre Environnement de Formation. Naturellement, ce langage est donc destiné à être utilisé par des personnes possédant des compétences en programmation. Dans le cas où les personnes responsables de la création des sessions de formation (i.e. les formateurs) ne possèdent pas de telles compétences, ceci constitue un problème. En effet, dans le cadre de nos travaux, nous souhaitons que ces personnes soient directement impliquées dans le processus de modélisation des scénarios, car ce sont elles qui possèdent l'expertise liée aux opérations métier à effectuer et à la pédagogie requise pour enseigner correctement ces opérations.

Dans ce chapitre, notre objectif est donc de proposer une méthodologie de conception en adéquation avec les compétences des formateurs, leur permettant ainsi de créer les scénarios qu'ils souhaitent de manière adaptée. Comme nous le présentons sur la Figure 15, cette méthodologie implique l'utilisation d'un environnement auteur au travers duquel les formateurs seront en mesure de modéliser leurs scénarios en utilisant un langage graphique intermédiaire qui devra ensuite être exporté vers notre langage dédié. De plus, l'environnement auteur mettra à leur disposition un ensemble de fonctionnalités visant à améliorer le processus de création de ces scénarios : interactions avec l'Environnement de Formation, réutilisation de précédents scénarios, etc.

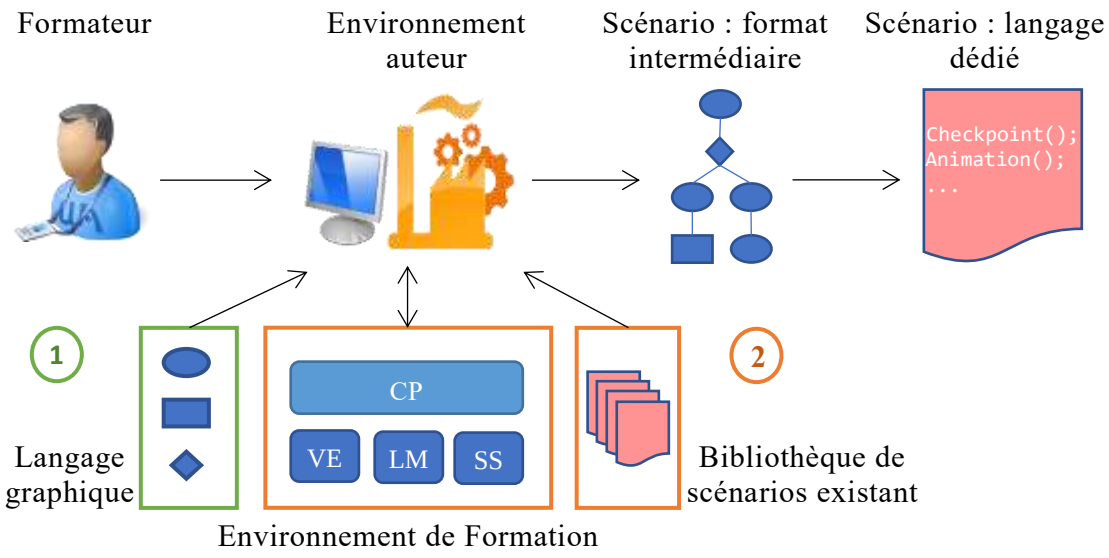


FIGURE 15. PROCESSUS DE CREATION DES SCENARIOS

4.2 Représentation graphique

Le premier point important de notre méthodologie concerne l'utilisation d'un langage graphique spécifique permettant à un formateur de décrire les scénarios dans un format intermédiaire adapté à leurs compétences et à leur expertise. Dans la partie 4.2.1, nous nous attachons donc à présenter les différentes dimensions de cette représentation et comment elles interagissent entre elles, au travers de l'exemple présenté en 3.3.2. Puis, nous nous intéressons en 4.2.2 au processus de conversion du format intermédiaire vers notre langage dédié, conversion qui est nécessaire afin de pouvoir exécuter des leçons dans notre Environnement de Formation.

4.2.1 Une représentation à plusieurs dimensions

Afin d'être en adéquation avec les compétences du formateur, le langage graphique que nous allons définir dans cette partie propose plusieurs niveaux de représentation associés aux trois grandes dimensions d'une leçon : le déroulement des activités, les objectifs et retours pédagogiques, et les éléments ludiques liés à la scénarisation des activités. En effet, les concepts liés à chacune de ces dimensions étant différents par nature, il ne nous paraît pas pertinent ni évident de les traiter de la même manière. C'est pourquoi nous souhaitons que notre langage graphique établisse une claire distinction entre ces dimensions, que nous présenterons respectivement en 4.2.1.1, 4.2.1.2 et 4.2.1.3. Cependant, toutes ces dimensions devant être liées pour former le scénario final dans le langage dédié associé, nous présenterons dans la partie 4.2.1.4 les caractéristiques de notre langage permettant d'effectuer cette association.

4.2.1.1 Description de l'Activité

Afin de décrire le déroulement des activités et donc un séquencement d'opérations, nous avons choisi de baser notre langage graphique sur le principe des Flowcharts (Tang, et al., 2008) (Araújo & Roque, 2009) et des blocs graphiques configurables (Van Est, Poelman, & Bidarra, 2011) tout en reprenant certains concepts et certaines notations introduites par BPMN (White, 2004) et qui sont adaptées à des personnes expertes dans un domaine métier particulier. Au travers de ce langage, notre objectif est de proposer un ensemble de blocs graphiques prédéfinis, que l'auteur pourra ensuite instancier autant que nécessaire pour former une représentation graphique qui définit l'enchaînement des opérations via l'utilisation de transitions entre les blocs.

La Figure 16 montre le modèle général que nous proposons afin de décrire la dimension Activités de nos scénarios. Comme évoqué précédemment, une de nos principales motivations dans la conception de ce modèle était de pouvoir fournir aux formateurs une liste prédéfinie de blocs utilisables dans différents contextes et qui soient en plus faciles à comprendre et à manipuler. Ainsi, nous définissons chaque bloc par :

- Un ***nom***, qui permet au formateur de rapidement comprendre son utilité. Ex : *Attendre que l'apprenant effectue une interaction*. Cette propriété est traductible puisqu'elle est visible par l'auteur des scénarios.
- Un ***label***, qui permet au formateur de comprendre l'utilité du bloc lorsqu'il est instancié pour décrire une partie des activités du scénario. Ce label peut-être dépendant des propriétés du bloc (voir plus bas). Ex : *Attendre que l'apprenant effectue l'interaction GARAGE.BUMPER.TAKE_OFF*. Cette propriété est aussi traductible.

- Une ou plusieurs ***catégorie***s, qui permettent de lui associer le ou les modules de l'Environnement de Formation avec lesquels il interagit. Par exemple, le bloc *Attendre que l'apprenant effectue une interaction* fait partie de la catégorie *3D*. Cela permet d'effectuer un premier niveau de classification des blocs dans la liste afin que le formateur puisse rapidement trouver celui qui l'intéresse. De plus, nous avons associé chaque catégorie à une couleur, afin de faciliter la compréhension de la représentation graphique de la description des activités.
- Un ***type***, permettant de définir le sens du bloc, c'est-à-dire la notion du langage naturel qu'il représente (voir 3.3.3.2). Par exemple, le bloc *Attendre que l'apprenant effectue une interaction* possède le type *Checkpoint* car il est associé à la notion de Point de Passage. Cela permet d'effectuer un deuxième niveau de classification dans la liste des blocs pour assister le formateur. Notons que certains types peuvent servir à définir plusieurs blocs décrivant différentes interactions avec l'Environnement de Formation (Checkpoint, Animation), alors que les autres sont liés à des blocs spéciaux et communs à l'ensemble des scénarios (StartActivity, EndActivity, Branching, Or, And). De plus, chaque type de scénario est associé à une forme spécifique, de manière à repérer aisément quelle est sa principale fonction.
- Une liste de ***propriétés***, qui permettent de lui associer un certain nombre d'informations en fonction du contexte où il est utilisé. Par exemple, le bloc *Attendre que l'apprenant effectue une interaction* possède une propriété qui représente l'identifiant de l'interaction souhaitée. De cette manière, il est possible de décrire avec ce même bloc le fait que le joueur doive démonter le pare-chocs puis poser la plaque sur le radar (en reprenant l'exemple de la partie 3.3.2). Cela permet ainsi de réduire le nombre de blocs nécessaires pour décrire l'ensemble des activités d'un scénario, et surtout d'avoir un ensemble fixe de blocs pour l'ensemble des scénarios se déroulant dans le cadre d'un domaine métier particulier.

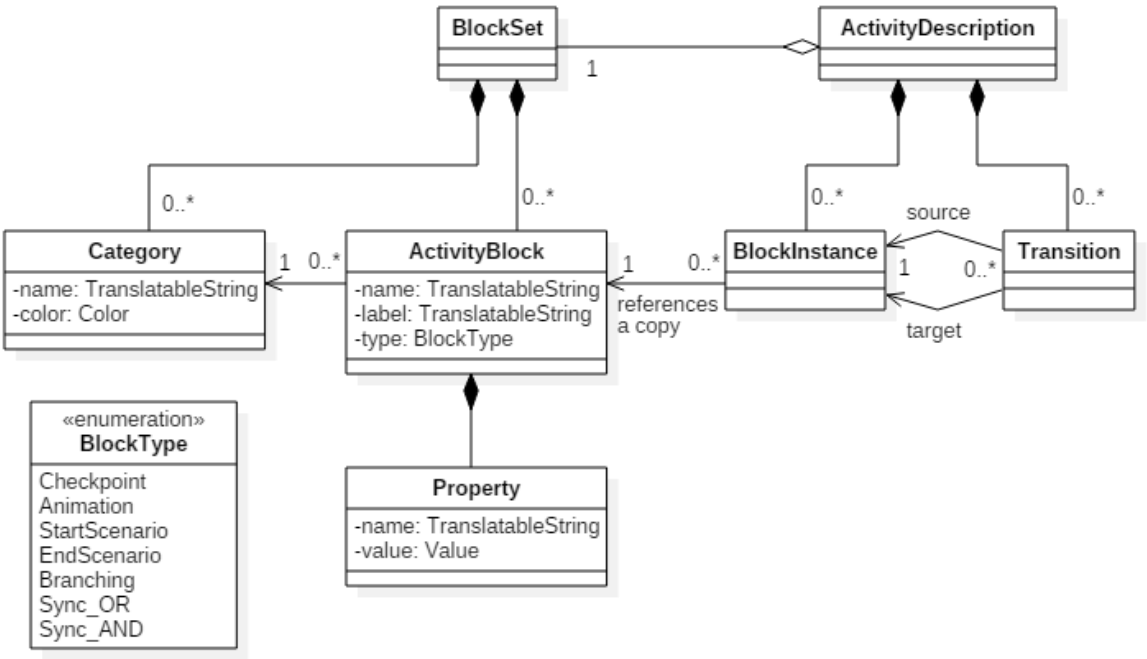


FIGURE 16. MODELE DE REPRESENTATION GRAPHIQUE DE L'ACTIVITE D'UN SCENARIO

Chaque instanciation de bloc est ensuite liée à une ou plusieurs transitions entrantes et sortantes, permettant ainsi de connaitre ses prédécesseurs et ses successeurs. Finalement, en utilisant cette représentation, la description du déroulement des activités de la leçon présentée en 3.3.2 se note comme sur la Figure 17 (la liste des blocs nécessaires est donnée en Annexe 3) :

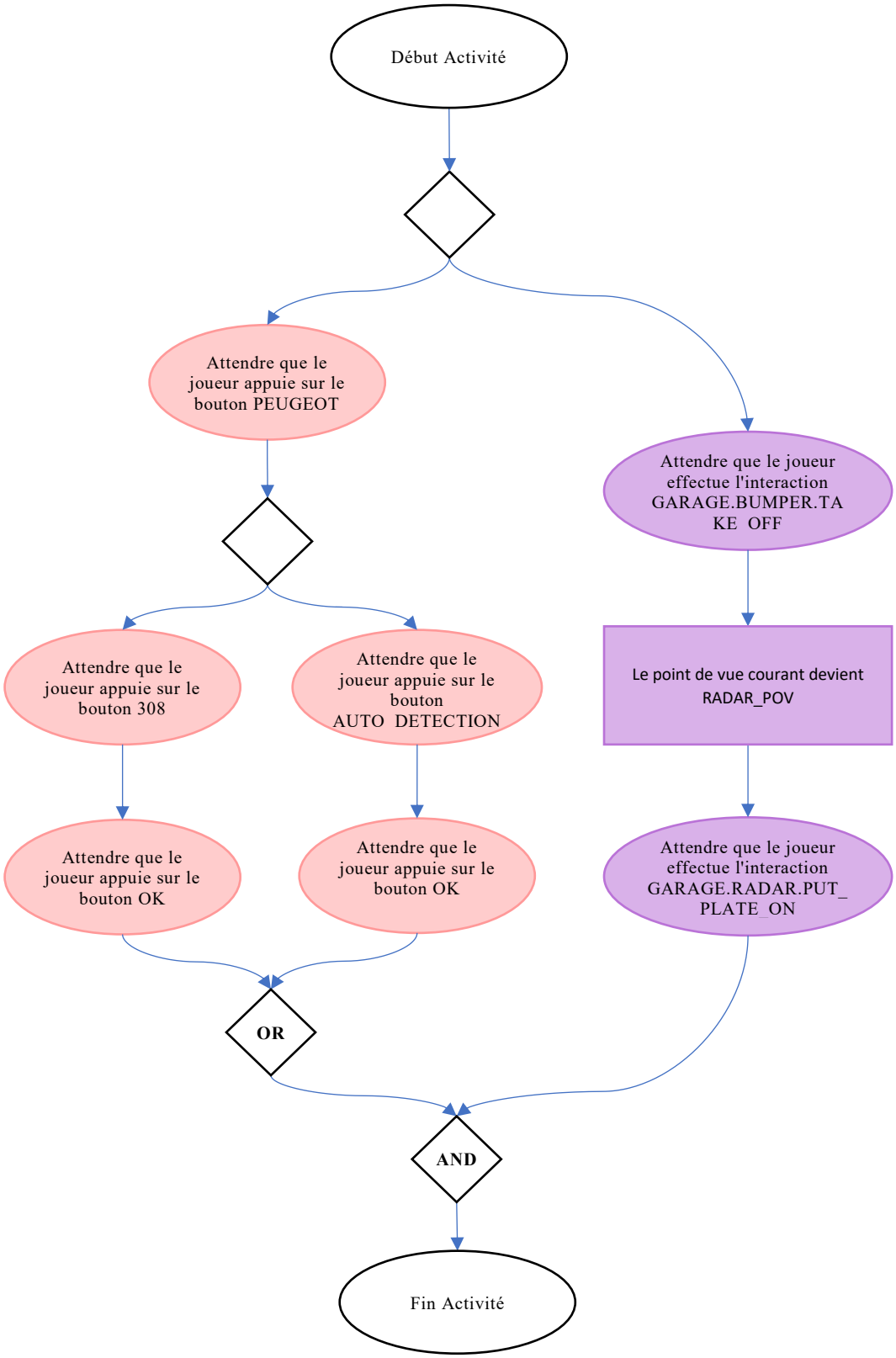


FIGURE 17. EXEMPLE DE DESCRIPTION DU DEROULEMENT DES ACTIVITES

4.2.1.2 Description de la Pédagogie

Afin de décrire la pédagogie, notre langage graphique propose un deuxième niveau de représentation avec un second formalisme. Pour cette deuxième dimension, nous avons observé dans le 0 que les éléments à décrire dans le cadre des scénarios concernent les objectifs pédagogiques et les retours pédagogiques. Dans cette partie, nous introduisons en plus la notion d'activité pédagogique, qui permet de lier ces deux concepts à un élément pivot qui sera plus tard utilisé pour associer la description de la pédagogie avec la description de l'Activité. La Figure 18 montre le modèle que nous avons conçu afin de représenter graphiquement ces différents éléments pédagogiques.

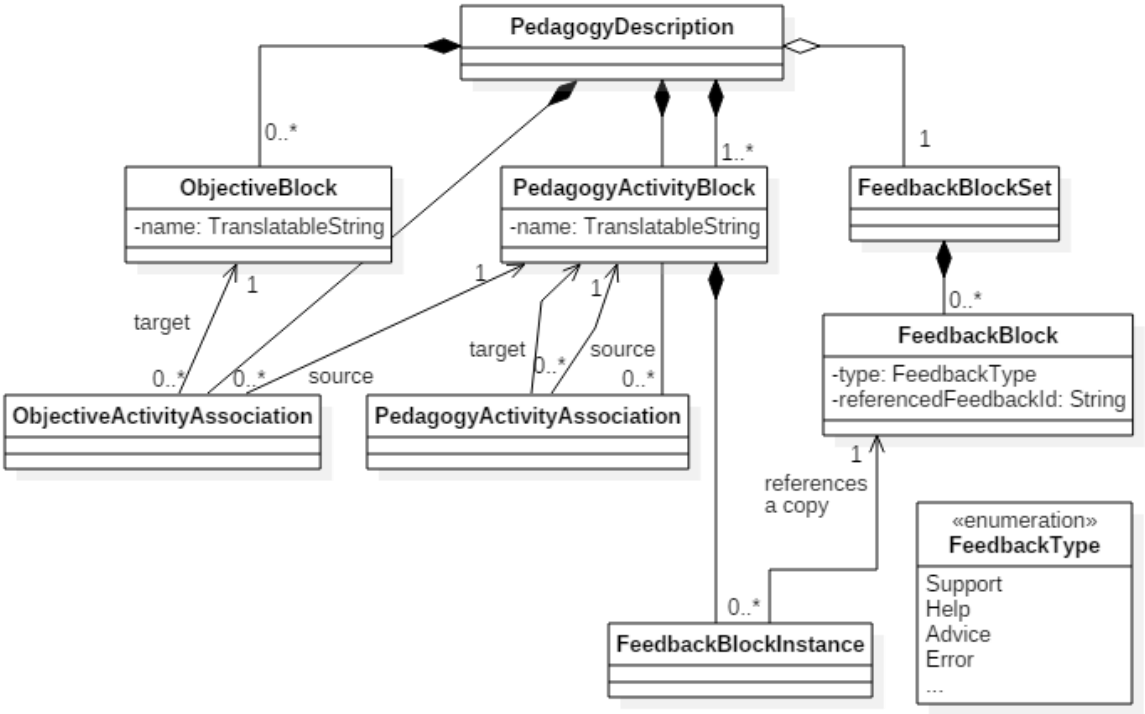


FIGURE 18. MODELE DE REPRESENTATION GRAPHIQUE DE LA PEDAGOGIE D'UN SCENARIO

La dimension pédagogique est donc définie par :

- Les ***blocs d'objectifs pédagogique***s. Ici, les éléments dont nous avons besoin sont uniquement un label traductible pour que l'auteur puisse identifier ces blocs, et la liste des activités pédagogiques auxquelles ils sont liés.
- Les ***blocs de retours pédagogique***s. De la même manière que pour les blocs permettant de décrire l'Activité, nous avons souhaité que ces éléments soient réutilisables dans différents contextes. En plus d'un type permettant de les classifier (ex : message d'aide, message d'erreur, etc.), les retours pédagogiques sont définis par une propriété contenant l'identifiant du retour qui devra effectivement être exécuté lorsque l'apprenant effectuera le scénario. Enfin, chaque type de retour pédagogique est associé à une icône caractéristique qui constitue le seul élément utilisé dans la représentation graphique de la Pédagogie afin de ne pas surcharger sa lisibilité.
- ***Les blocs d'activités pédagogique***s. Conceptuellement, ces activités représentent des séquences d'opérations importantes d'un point de vue pédagogique, qui serviront en suite

de base pour que l'outil de formation puisse fournir un compte-rendu de la session à la fin du scénario. Dans le cadre de la description de la Pédagogie uniquement, les propriétés intéressantes sont un label traductible permettant de leur associer une signification compréhensible par le formateur, les objectifs pédagogiques associés, et les instances des retours pédagogiques associés. Les activités pédagogiques représentant des séquences d'action importantes parmi les activités métier enseignées, nous avons considéré qu'il était important d'être en mesure d'exécuter certains retours pédagogiques à leur démarrage (ex : message d'accompagnement, message d'aide), et d'autres à leur validation (ex : quiz). C'est pourquoi notre modèle fait la distinction entre les *pre-feedbacks* et les *post-feedbacks*.

Graphiquement, les objectifs pédagogiques sont représentés par un bloc rectangulaire de couleur verte, les activités pédagogiques par un bloc rectangulaire de couleur blanche, et les instances de retours pédagogiques par l'icône correspondant à leur type respectif. Chaque icône est de plus étiquetée avec l'identifiant du retour devant effectivement être exécuté. Ensuite, l'association entre les activités pédagogiques et les objectifs pédagogiques se représente à l'aide d'une transition entre les deux blocs correspondants. Enfin, l'association entre les activités pédagogiques et les retours pédagogiques s'effectue en plaçant directement sur le bloc de l'activité les icônes correspondant aux types des retours souhaités (en haut à gauche pour ceux à exécuter au démarrage de l'activité, en bas à droite pour ceux à exécuter à sa validation).

Finalement, la description de la Pédagogie de l'exemple présenté dans partie 3.3.2 se note comme illustré sur la Figure 19 en suivant notre modèle et notre représentation graphique (les différents types de retours pédagogiques utilisés sont décrits en Annexe 3) :

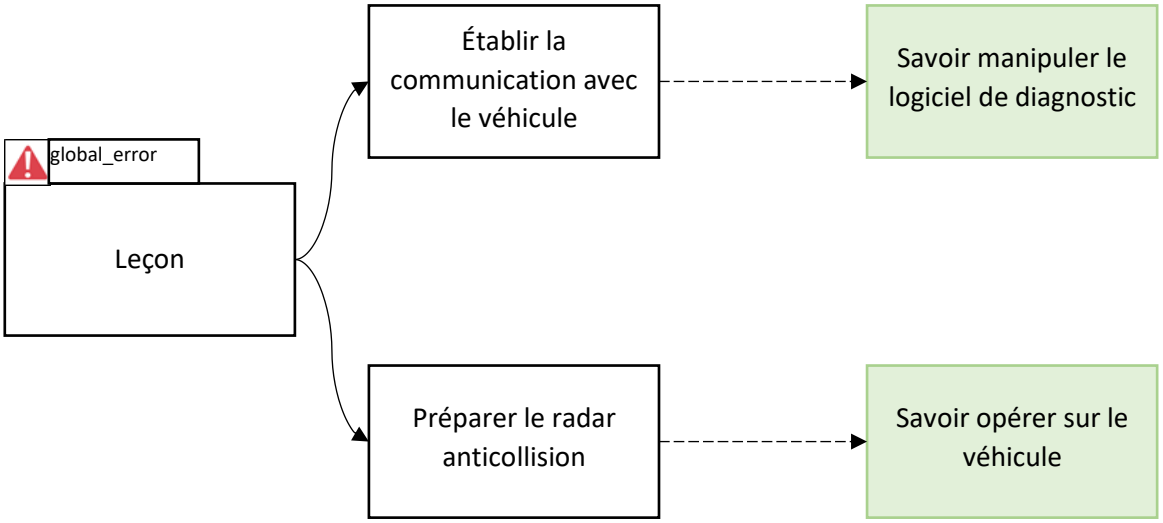


FIGURE 19. EXEMPLE DE DESCRIPTION DE LA DIMENSION PEDAGOGIQUE

Notons le bloc d'activité pédagogique "Leçon" qui existe pour tout scénario et qui correspond au déroulement de l'ensemble de la leçon étudiée.

4.2.1.3 Description des éléments ludiques

La dernière dimension des scénarios à décrire concerne l'ensemble des concepts ludiques qui y sont liés, et permettant de scénariser et de mettre en contexte la leçon concernée. Ainsi, nous avons conçu un troisième format de représentation (comparable à un arbre) pour décrire

graphiquement la notion de mission, de sous-mission, et de mise en scène. De la même manière que pour la description de l'Activité et de la Pédagogie, ce format suit un modèle spécifique que nous avons défini et que nous présentons sur la Figure 20.

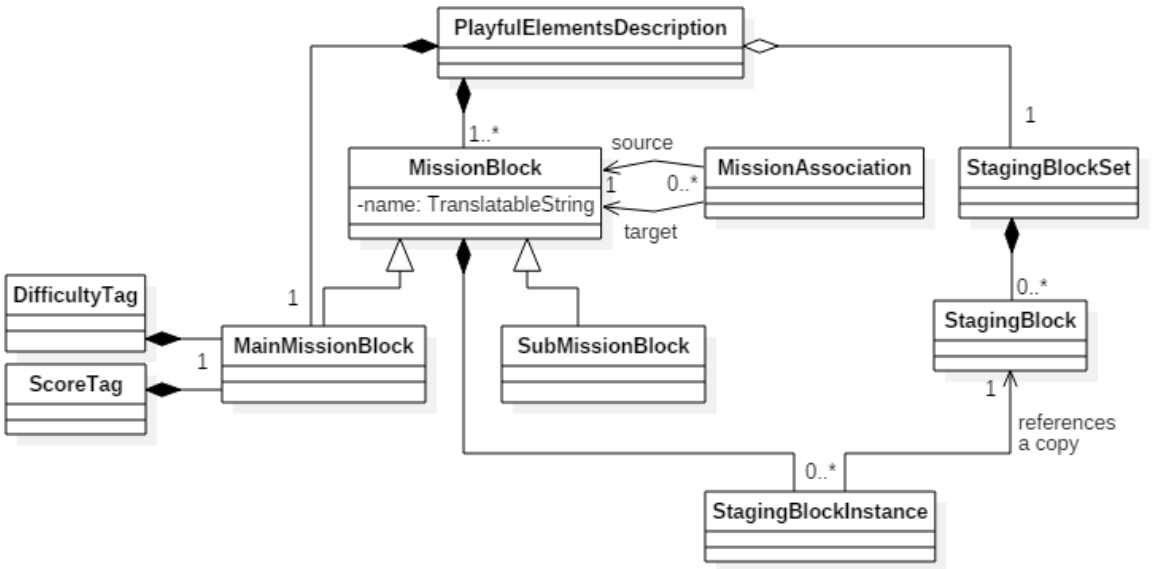


FIGURE 20. MODELE DE REPRESENTATION DES ELEMENTS LUDIQUES D'UN SCENARIO

Les différentes composantes du format de représentation graphique des éléments ludiques sont les suivantes :

- Le ***bloc de mission***. Il possède un identifiant interne ainsi qu'un intitulé afin que l'auteur puisse identifier l'objectif principal de la leçon. Il est associé à une étiquette de difficulté, à une étiquette de score, et à des sous-missions via une association particulière. Nous le représentons graphiquement à l'aide d'un bloc rectangulaire de couleur orange pâle dont le label correspond à son intitulé.
- L'***étiquette de difficulté***. Son unique propriété est un entier représentant la difficulté de la mission. Nous le représentons à l'aide d'une étiquette attachée au bloc de mission, et dont le label affiche "*difficulté = [difficulty]*.
- L'***étiquette de score***. Son unique propriété est un entier représentant le score requis pour valider la mission sous forme de pourcentage. Nous le représentons à l'aide d'une étiquette attachée au bloc de mission, et dont le label affiche "*score requis = [score]%*".
- Les ***blocs de sous-mission***. Ils possèdent un identifiant interne ainsi qu'un intitulé afin que l'auteur puisse identifier l'objectif associé à cette sous-mission. Ils peuvent être liés à d'autres blocs de sous-mission enfant via une association spécifique, et leur parent est soit un bloc de sous-mission, soit de bloc de mission. De plus, ils peuvent être associés à des blocs d'instances de mise en scène via l'association correspondante. En suivant le même raisonnement que pour les blocs d'activité pédagogique qui représentent des séquences d'opération importantes dans la leçon, nous avons choisi de faire la distinction entre les mises en scène devant s'effectuer au démarrage de la sous-mission, et celles devant s'effectuer à sa complétion. Graphiquement, nous les représentons avec un bloc rectangulaire de couleur blanche.

- Les ***associations mission/sous-missions***. Elles permettent d'associer à la mission principale autant de sous-missions que nécessaire. Ses deux propriétés sont donc le bloc de mission parent et le bloc de sous-mission enfant associé, que nous représentons à l'aide d'une simple transition entre les deux.
- Les ***associations sous-mission/sous-missions***. Similairement aux associations mission/sous-mission, elles permettent d'associer à chaque sous-mission autant de sous-missions que nécessaire. Ses deux propriétés sont donc le bloc de sous-mission parent et le bloc de sous-mission enfant, que nous représentons à l'aide d'une transition entre les deux.
- Les ***blocs d'instance de mise en scène***. De la même manière que pour les blocs d'instance d'activité et aux blocs d'instance de retour pédagogique, nous avons souhaité proposer une bibliothèque de blocs de mise en scène qui soient instanciables afin de pouvoir être utilisés dans différents contextes. Un bloc d'instance de mise en scène instancie donc un bloc de mise en scène, qui lui est défini par un type permettant de le classifier (ex : dialogue, vidéo) et une propriété contenant l'identifiant de la mise en scène qui devra effectivement être exécutée lorsque la leçon sera effectuée par l'apprenant. Nous les représentons graphiquement par un bloc contenant l'icône du type de la mise en scène, et un label affichant l'identifiant associé.
- Les ***compositions mission/blocs d'instance de mise en scène***. Elles permettent de lier des mises en scène à la mission principale ou à des sous-missions particulières. Ses propriétés sont donc le bloc de mission parent et le bloc d'instance de mise en scène enfant, que nous représentons graphiquement en superposant le deuxième bloc sur le premier (en haut à gauche pour les mises en scène devant s'effectuer au démarrage de la sous-mission, en bas à droite celles devant s'effectuer à sa complétion).

Finalement, la description des éléments ludiques de l'exemple présenté dans la partie 3.3.2 se représente graphiquement comme illustré sur la Figure 21 (les différentes mises en scènes utilisées sont décrites en Annexe 3) :

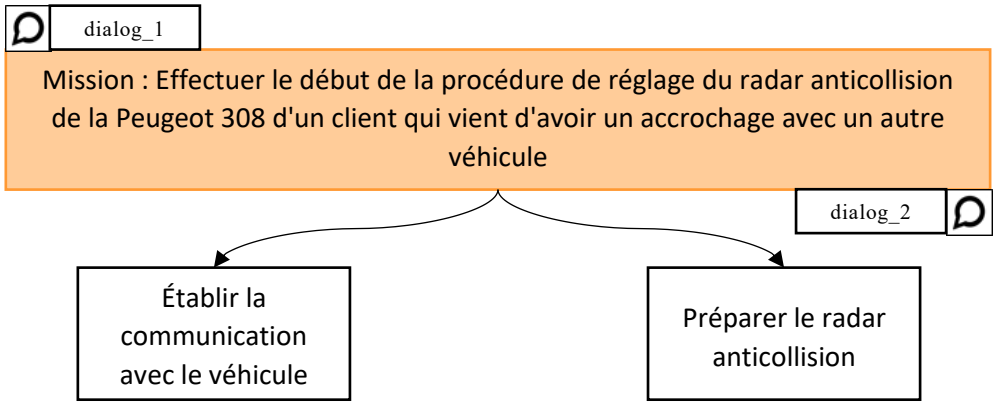


FIGURE 21. EXEMPLE DE DESCRIPTION DE LA DIMENSION LUDIQUE

Nous notons alors que les sous-missions définies ici correspondent aux activités pédagogiques définies dans la partie 4.2.1.2. Bien que la distinction soit bien réelle (les sous-missions existent pour scénariser la leçon, alors que les activités pédagogiques sont là pour identifier les opérations importantes de la leçon liées à des objectifs pédagogiques particuliers), ces deux concepts seront souvent liés du fait qu'ils se rapportent tous deux à des séquences

d'opérations importantes du scénario. Pour cette raison, l'environnement auteur pourra permettre de générer les parties correspondantes de la dimension pédagogique à partir de la dimension ludique (et vice-versa).

4.2.1.4 Association des dimensions

Une fois les trois dimensions du scénario décrites chacune dans leur propre format, il reste à faire l'association entre elles. En effet, de la même manière que notre langage dédié unifie ces dimensions dans une seule et même représentation, nous avons besoin d'établir le lien entre les progrès du joueur dans la leçon et les différents éléments pédagogiques et ludiques. De manière assez logique, nous avons donc choisi d'effectuer cette association au niveau de la description du déroulement de la leçon.

Dans les dimensions pédagogiques et ludiques, nous avons défini deux éléments (respectivement, activités pédagogiques et sous-missions) qui se rapportent conceptuellement à des séquences d'opérations importantes de la leçon, et qui doivent donc par nature être liées avec la description de l'Activité. Afin de représenter ce lien, nous introduisons un élément pivot dans notre représentation graphique de l'Activité, la sous-activité, qui permet de découper la description globale de l'Activité en plusieurs sous-séquences d'opérations. Ces séquences peuvent ensuite être rattachées aux éléments correspondants des deux autres dimensions. La Figure 22 montre les éléments ajoutés et modifiés par rapport aux modèles présentés dans les parties 4.2.1.1, 4.2.1.2 et 4.2.1.3 :

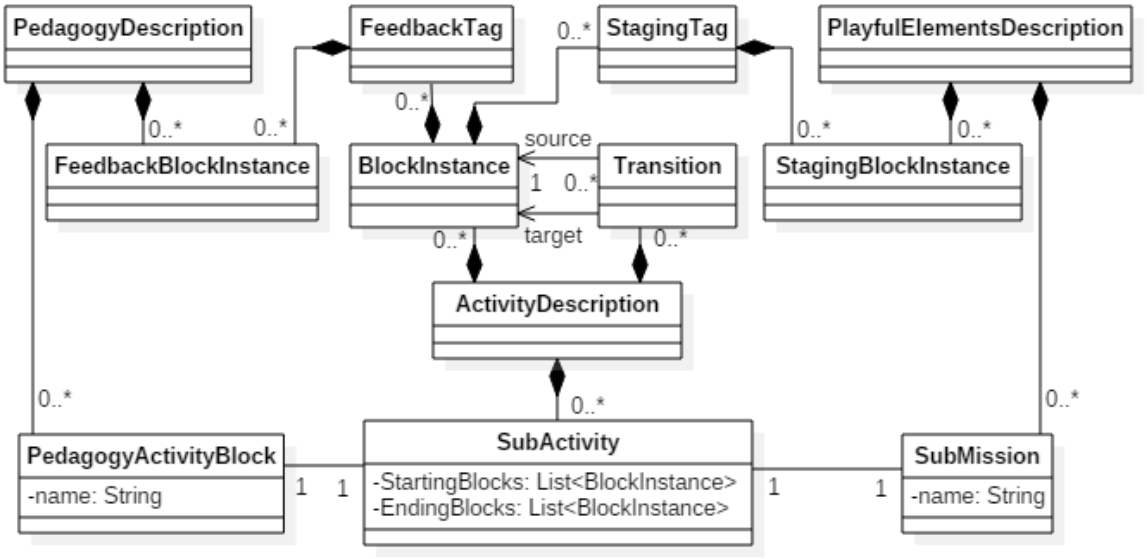


FIGURE 22. ÉLEMENTS AJOUTES ET MODIFIES AUX DIFFERENTS MODELES DE REPRESENTATION GRAPHIQUE DE SCENARIO AFIN DE LIER LES DIMENSIONS ENTRE ELLES

En plus de l'activité pédagogique et de la sous-mission associée, les sous-activités sont définies par un ou plusieurs blocs de début, et un ou plusieurs blocs de fin, qui marquent donc respectivement le démarrage et la fin de ces sous-activités. Graphiquement, elles sont représentées par des blocs en pointillé qui viennent entourer les blocs d'activités formant la séquence correspondante. De plus, un label affiche les intitulés de l'activité pédagogique et de la sous-mission associée.

De plus, nous avons aussi souhaité pouvoir associer à chaque opération de la description de l'Activité un certain nombre de retours pédagogiques et d'éléments de mise en scène. Cela est par exemple utile si le formateur souhaite spécifier pour chaque opération unitaire une aide indiquant comment accomplir la prochaine action attendue, sans avoir à définir une sous-activité n'encapsulant que l'instance de bloc concernée. Pour cela, chaque instance de bloc peut maintenant posséder une étiquette de retour pédagogique et une étiquette de mise en scène, qui elles-mêmes peuvent être associées aux éléments pédagogiques et ludiques correspondants. Afin de rester cohérent avec les missions et les activités pédagogiques, ces retours pédagogiques et éléments de mise en scène peuvent être liés en tant que pré-action ou post-action, permettant ainsi de spécifier l'instant où ils seront pris en compte par rapport à l'exécution du bloc d'activité associé. Graphiquement, ces étiquettes se représentent avec une icône particulière à côté du bloc associé, complétée par les éléments qui leur sont attachés (bloc de retour pédagogique ou bloc de mise en scène).

Finalement, la représentation graphique de l'Activité (voir partie 4.2.1.1) est complétée comme illustré sur la Figure 23. Nous notons que des étiquettes de retour pédagogique et de mise en scène sont attachées aux blocs de début et de fin de l'Activité. En effet, ils correspondent aux retours pédagogiques et éléments de mises en scène spécifiés dans les dimensions respectives (activité pédagogique "Leçon" et bloc de mission principale). De manière générale, les retours pédagogiques et éléments de mise en scène spécifiés dans les dimensions respectives doivent se retrouver via une étiquette sur les instances de blocs de la description de l'Activité correspondantes. Ces blocs sont alors les blocs de début / de fin de la sous-activité faisant le lien avec les blocs d'activité pédagogique et de mission/sous-mission auxquels ils sont associés. Enfin, nous observons qu'une étiquette de retour pédagogique est associée à chaque bloc de *Checkpoint* afin de pouvoir leur associer une aide indiquant comment réaliser l'interaction associée (tel que spécifié dans l'exemple de leçon en 3.3.2). Nous n'avons pas spécifié le retour effectivement associé pour ne pas surcharger la représentation graphique globale, mais le principe est le même que pour les blocs de début et de fin de l'Activité.

4.2.2 Conversion vers le langage dédié

Une fois le scénario décrit en utilisant notre représentation graphique, il reste malgré tout nécessaire de le convertir dans notre langage dédié afin de pouvoir être exécutable par notre outil de formation. Cette conversion sera bien entendu effectuée directement par l'environnement auteur sans intervention de la part du formateur. Dans cette partie, nous nous intéressons donc à décrire cette phase de conversion, avec les deux grandes phases qui lui sont associées et qui assurent que l'opération se déroule correctement. Dans un premier temps, nous décrirons un certain nombre de règles permettant de valider la représentation graphique du scénario, et particulièrement la description de l'Activité, afin de s'assurer de sa cohérence "syntaxique" (4.2.2.1). Dans un second temps, nous présenterons les règles de transformations qui permettent de convertir à proprement parler le scénario depuis sa représentation graphique vers sa représentation en langage dédié (4.2.2.2).

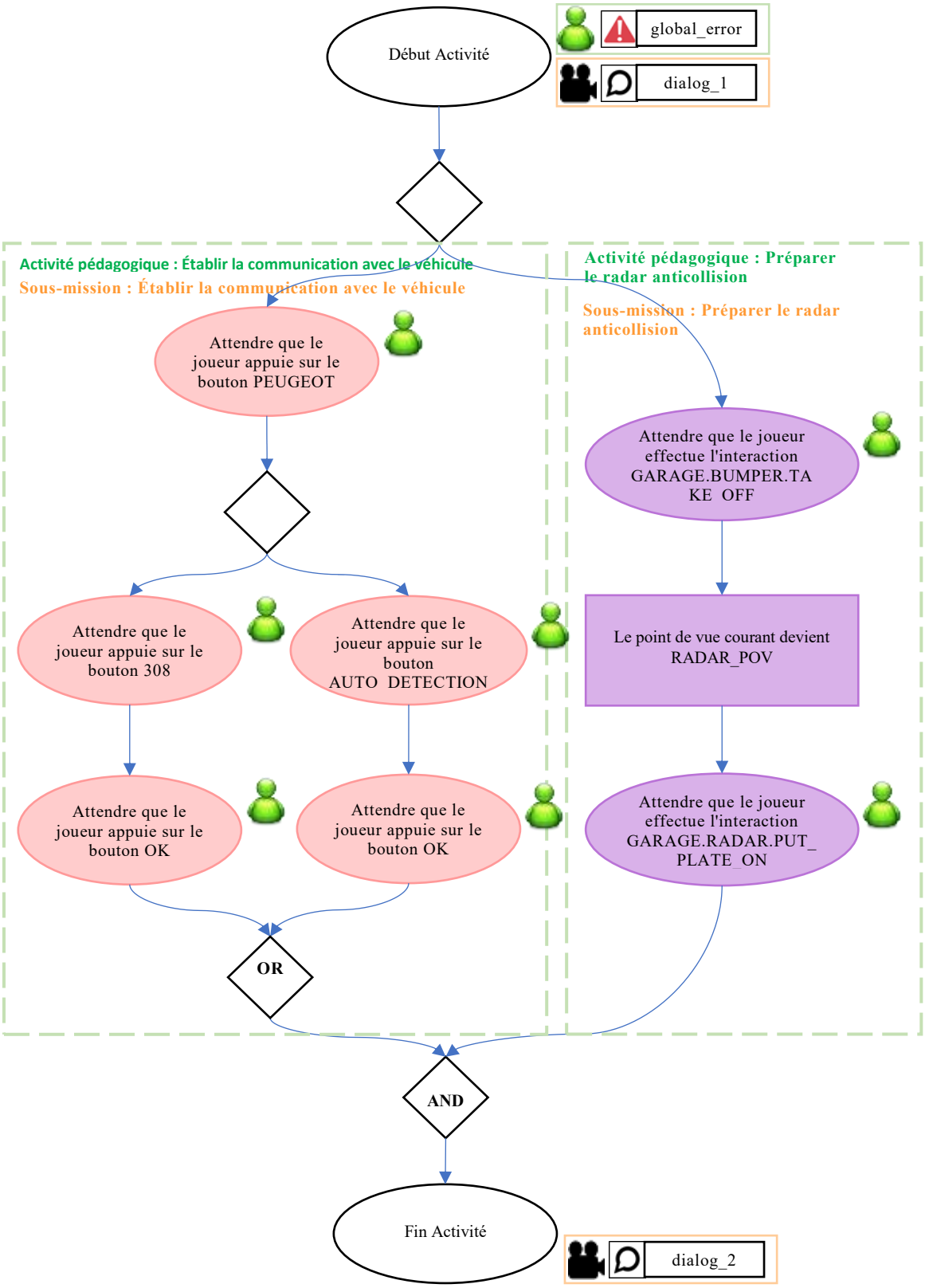


FIGURE 23. EXEMPLE DE DESCRIPTION DU DEROULEMENT DES ACTIVITES COMPLETEE AVEC L'ASSOCIATION DES DIMENSION PEDAGOGIQUES ET LUDIQUES

4.2.2.1 Validation

Afin d'être en mesure de générer la représentation en langage dédié associée à la représentation graphique du scénario, nous devons tout d'abord nous assurer de sa validité "syntaxique", c'est-à-dire du respect de certaines règles liées aux différents éléments graphiques introduits dans la partie 4.2.1. Nous présentons ci-dessous les règles qui viennent s'ajouter à celles induites par les différents diagrammes de classe et que nous avons donc déjà évoquées dans la section correspondante.

Dans un premier temps, la partie du scénario la plus importante à valider concerne la description de l'Activité. En effet, les autres dimensions gravitant autour du déroulement des opérations, il est tout d'abord nécessaire de vérifier que les séquences de blocs décrites sont correctes. Pour cela, la méthodologie que nous proposons invite à découper cette dimension en plusieurs éléments :

- Séquence d'activité. C'est l'élément central qui décrit un enchaînement d'opérations dans l'Environnement de Formation. Plus précisément, une séquence d'activité est une suite stricte de blocs d'instruction et de structures de contrôle (voir points suivants), et n'autorise ni les boucles, ni les embranchements.

  Tout scénario possède au minimum une Séquence d'activité, que nous appelons séquence principale, et qui doit obligatoirement :

  - commencer par une instance de bloc de type *StartActivity*. Cela est en effet nécessaire afin de définir le point de départ de la description de l'Activité.
  - se terminer via une instance de bloc de type *EndActivity* afin de définir la fin de la description de l'Activité. Rappelons qu'il est possible de définir des fins alternatives pour notre scénario, qui seront alors décrites en utilisant une instance de ce bloc dans plusieurs "sous-séquences d'activité" définies dans le cadre des Structures de Contrôle (voir dernier point).
- Bloc d'instruction. C'est l'élément unitaire de plus bas niveau dans le découpage que nous effectuons ici. Il se rapporte simplement aux instances de bloc d'activité de type *Checkpoint*, *Animation*, *StartActivity*, et *EndActivity*. Ces blocs d'instructions doivent posséder un unique élément prédécesseur (bloc d'instruction ou structure de contrôle) et un unique élément successeur (bloc d'instruction ou structure de contrôle), excepté pour :
  - les instances de bloc de type *StartActivity* qui ne doivent avoir aucun prédécesseur ;
  - les instances de bloc de type *EndActivity* qui ne doivent avoir aucun successeur.
- Structure de contrôle. C'est l'élément qui permet de déclarer des "sous-séquences d'activité", via l'utilisation d'une instance de bloc de type *Branching*. Une structure de contrôle doit respecter les points suivants :
  - Le bloc d'embranchement doit avoir un unique élément prédécesseur (bloc d'instruction ou structure de contrôle), mais peut avoir plusieurs éléments successeurs (minimum 1) de type bloc d'instruction et qui constituent le premier bloc de chaque séquence déclarée par la structure de contrôle.
  - Le premier bloc de chaque séquence déclarée doit être une instance de bloc de type *Checkpoint*. Cela est en effet nécessaire afin définir la condition de démarrage de chaque séquence.

- Chaque séquence déclarée doit posséder la même instance de bloc de synchronisation (*Sync*_OR ou *Sync*_END) OU se terminer par une instance de bloc de type *EndActivity*. Il est ainsi possible qu'une structure de contrôle n'ait aucun bloc de synchronisation.
- S'il existe, le bloc de synchronisation doit avoir un unique élément successeur (bloc d'instruction ou structure de contrôle), mais peut avoir plusieurs éléments prédécesseurs (correspondant au dernier élément de chaque séquence déclarée par la structure de contrôle, pouvant être de type bloc d'instruction ou structure de contrôle).
- S'il existe, le bloc de synchronisation est unique à la structure de contrôle associée et n'est utilisable que par les séquences qu'elle déclare. Autrement dit, plusieurs structures de contrôle ne peuvent pas avoir le même bloc de synchronisation.

La description graphique de l'Activité est validée uniquement s'il est possible de découper strictement la description graphique de l'Activité en ces trois éléments (et en respectant l'ensemble des règles associées). Dans le cas contraire, l'environnement auteur devra remonter à l'utilisateur un message indiquant à quel niveau les erreurs apparaissent (soit en temps réel au moment de l'édition, soit au moment de l'exportation). La Figure 24 ci-dessous illustre le découpage de l'Activité en reprenant la représentation obtenue à la fin de la partie 4.2.1.1.

Dans un second temps, il est nécessaire de valider la représentation graphique associée à description de la Pédagogie. Pour cela, il suffit de vérifier les points suivants :

- Les associations entre les blocs d'activité pédagogiques doivent définir un arbre. Ainsi, la hiérarchie des blocs d'activité pédagogique doit commencer à partir d'un unique bloc racine. Chaque nœud peut ensuite posséder autant de blocs enfant que souhaité, mais doit posséder un seul et unique bloc parent (sauf le bloc racine qui n'en possède aucun).
- Une association entre bloc d'activité pédagogique et bloc d'objectif pédagogique est valide uniquement si le bloc d'activité parent est lui aussi associé au bloc d'objectif pédagogique. Si ce bloc d'activité parent ne spécifie aucune association, il faut alors effectuer cette vérification sur son propre bloc d'activité parent (et ainsi de suite jusqu'à parvenir à un bloc d'activité qui spécifie une association avec au moins un bloc d'objectif pédagogique). Si l'on remonte de cette manière jusqu'au bloc racine pour effectuer cette vérification et que celui-ci ne spécifie aucune association avec des blocs d'objectif pédagogique, alors l'association en cours de vérification est validée car l'activité pédagogique principale est associée implicitement à tous les objectifs pédagogiques.
- Les blocs de retour pédagogique doivent référencer un identifiant de retour pédagogique existant (qui devra donc avoir été précédemment défini dans la base de données pédagogique liée au scénario courant).

De la même manière, il faut valider la représentation graphique associée à la description des éléments ludiques en vérifiant les points suivants :

- Les associations entre les blocs de mission doivent définir un arbre dont le bloc racine est le bloc de mission principale. Chaque bloc de mission peut ensuite posséder autant de blocs enfant que souhaité, mais doit posséder un seul et unique bloc parent (mission principale ou sous-mission).

- Les blocs de mise en scène doivent référencer un identifiant de mise en scène existant (qui devra donc avoir été précédemment défini dans la base de données des éléments ludiques liée au scénario courant).

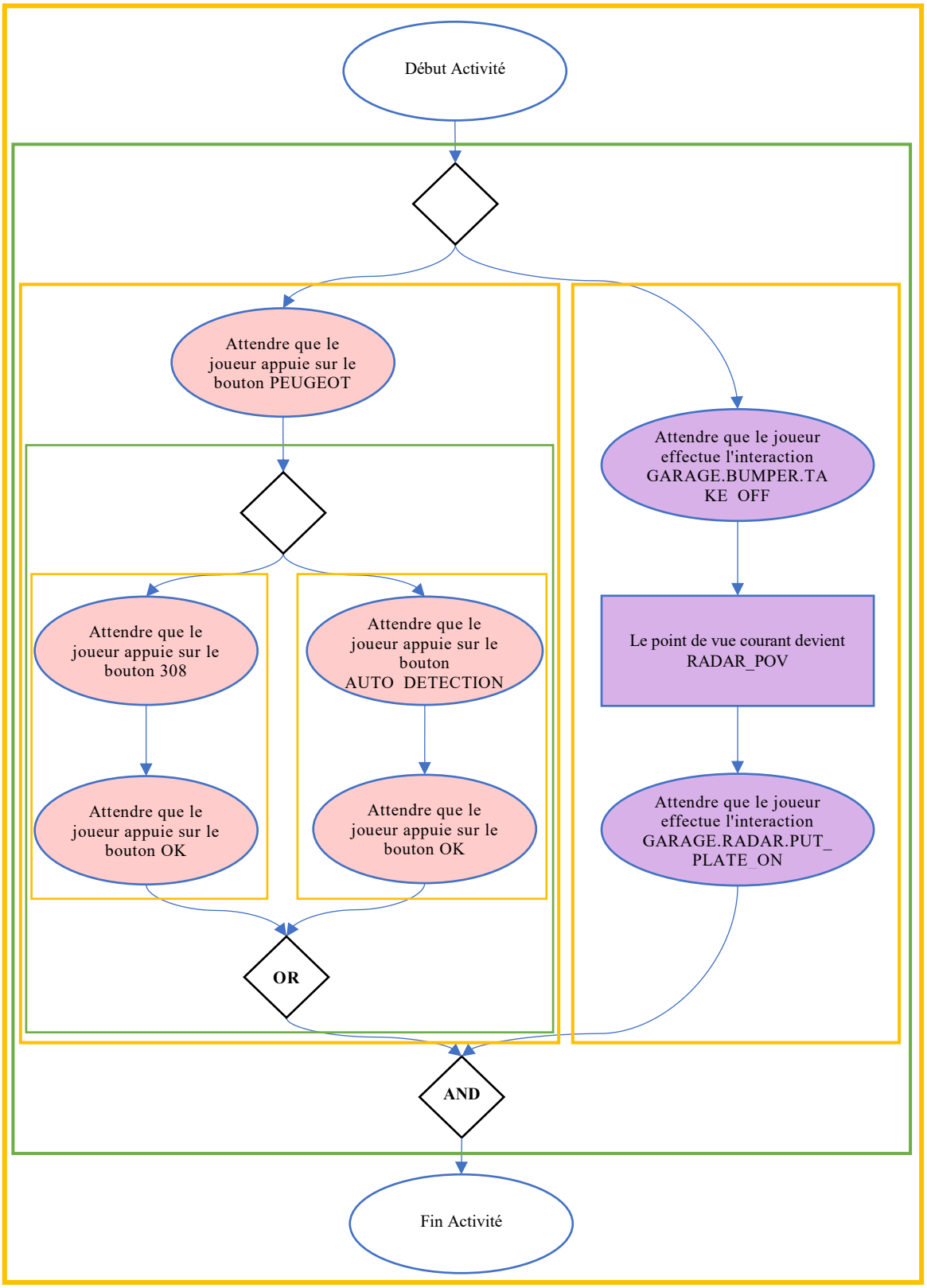


FIGURE 24. DECOUPAGE DE L'ACTIVITE (CONTOURS EN JAUNE POUR LES SEQUENCES D'ACTIVITE, EN VERT POUR LES CONTROLES DE STRUCTURE, ET EN BLEU POUR LES BLOCS D'INSTRUCTION)

Enfin, la dernière étape de cette phase de validation consiste à vérifier l'association entre ces dimensions. Pour cela, il suffit de s'assurer que chaque instance de bloc marquant le début d'une sous-activité mène à au moins une instance de bloc marquant la fin de cette même sous-activité (et vice-versa). Cela traduit le fait de vérifier qu'une sous-activité ne peut-être commencée que si elle se termine à un instant donné du scénario, et inversement qu'une sous-activité ne peut se terminer que si elle a été commencée précédemment durant le déroulement des opérations.

Finalement, nous considérons que la représentation graphique du scénario est validée si et seulement si l'ensemble des points associés aux différentes dimensions des leçons étudiées et décrits ci-dessus sont validés. Dès lors, nous pouvons nous intéresser à la phase suivante du processus de conversion : l'exportation.

4.2.2.2 Exportation

Une fois validé, le scénario doit être traduit depuis sa représentation graphique vers son équivalent utilisant le langage dédié défini dans la partie 3.3.3, afin de pouvoir finalement être exécuté par notre outil de formation. Nous appelons cette dernière étape du processus de conversion l'étape d'exportation. Pour ce faire, nous avons choisi d'utiliser directement le découpage de la description de l'Activité effectué lors de l'étape de validation, car il permet d'apporter à la représentation graphique du scénario les informations séquentielles requises pour décrire l'ensemble des éléments de notre langage dédié.

L'environnement auteur a alors en charge de traiter la description de l'Activité de manière chronologique, séquence d'activité par séquence d'activité, en commençant par la séquence d'activité principale. Ainsi, lors de l'exportation d'une séquence d'activité :

- Si nous rencontrons un bloc d'instruction, nous regardons d'abord s'il constitue un bloc de début ou un bloc de fin d'une des sous-activités de la description de l'Activité associée à une activité pédagogique ou à une sous-mission. En fonction de cette observation et des éléments auxquels fait effectivement référence la sous-activité :
    - Si c'est un bloc qui marque le début d'une sous-activité, nous déclarons dans l'ordre :
        - Une instruction `Pedagogical_Activity`, avec la liste des identifiants des objectifs pédagogiques associés à l'activité pédagogique concernée.
        - Les instructions `PedagogicalFeedback` associés au démarrage de la `Pedagogical_Activity`.
        - Une instruction `SubMission`, avec l'identifiant de la sous-mission concernée.
        - Les instructions `Staging` associés au démarrage de la `SubMission`.
    - Dans tous les cas, nous déclarons les instructions associées au bloc lui-même dans l'ordre suivant :
        - Les instructions `PedagogicalFeedback` associées aux pré-actions du bloc.
        - Les instructions `Staging` associées aux pré-actions du bloc.
        - La ou les instructions en langage dédié correspondantes au bloc même (`Checkpoint`, `Animation`, ou `End`).
        - Les instructions `Staging` associées aux post-actions du bloc.
        - Les instructions `PedagogicalFeedback` associées aux post-actions du bloc.
    - Si c'est un bloc qui marque la fin d'une sous-activité, nous déclarons dans l'ordre :

- Les instructions `Staging` associées à la validation de la `SubMission` associée.
- La fin de la `SubMission` associée.
- Les instructions `PedagogicalFeedback` associées à la validation de la `Pedagogical_Activity` associée.
- La fin de la `Pedagogical_Activity` associée.

- Si nous rencontrons une structure de contrôle, nous déclarons dans l'ordre :
  - Une instruction `Alternative_Activities` ou `Parallel_Activities` en fonction du type du bloc de synchronisation.
  - Les instructions `PedagogicalFeedback` associées au bloc d'embranchement.
  - Les instructions `Staging` associées au bloc d'embranchement.
  - Les différentes instructions `Branch` de la structure de contrôle, qui correspondent aux différentes séquences d'activités qui suivent le bloc d'embranchement. La condition d'entrée de chaque Branch correspond à la condition du bloc d'activité de type *Checkpoint* qui marque le début de chacune de ces séquences. Ensuite, nous exportons la séquence d'activité en ignorant donc ce premier bloc de type *Checkpoint* (et éventuellement les retours pédagogiques / mise en scène associés) normalement en suivant le processus global décrit dans le point précédent.
  - La fin de l'instruction Alternative_Activities / Parallel_Activities.
  - Les instructions `Staging` associées au bloc de synchronisation.
  - Les instructions `PedagogicalFeedback` associées au bloc de synchronisation.

Une fois la séquence d'activité principale traitée dans son ensemble, le processus d'exportation du scénario depuis la représentation graphique vers le langage dédié est terminé. En se basant sur l'exemple de la partie 3.3.2 et sa représentation graphique finale donnée en 4.2.1.4, nous retrouvons alors le scénario tel que nous l'avions défini à la fin de la partie 3.3.3.4, scénario qu'il est ensuite possible d'exécuter au travers de notre outil de formation.

4.3 Améliorer le processus de description du scénario

Décrire des scénarios par l'intermédiaire du langage graphique présenté en 4.2 représente une première grande partie de notre méthodologie visant à rendre le processus accessible aux formateurs. Cependant, la modélisation de ces scénarios devant toujours s'effectuer manuellement, on voit apparaitre plusieurs problématiques liées à la description de ses différentes dimensions. D'un côté, le processus de modélisation nécessite une bonne connaissance du modèle de l'Environnement de Formation afin d'identifier les propriétés et les valeurs associées qui permettent de décrire le déroulement de la leçon, et sort donc du cadre des compétences du formateur. D'un autre côté, en considérant des activités métier à enseigner plus longues et plus complexes que l'exemple donné en 3.3.2, le processus de description lui-même peut devenir très long, provoquant ainsi une augmentation de la probabilité pour le formateur d'effectuer des erreurs. Afin de répondre à ces problématiques et donc d'améliorer le processus de description global, nous proposons dans cette partie deux grandes fonctionnalités impliquant l'utilisation de l'Environnement de Formation (4.3.1) et la réutilisation de précédents scénarios (4.3.2).

4.3.1 Utiliser l'Environnement de Formation dans le processus

Afin de s'abstraire de la complexité de l'Environnement de Formation et de prévenir les erreurs durant la description des scénarios et plus particulièrement du déroulement de la leçon, la méthodologie que nous définissons dans ce chapitre propose d'utiliser directement l'EF dans le processus au travers de plusieurs formes d'interactions. Notre intérêt s'est donc tout d'abord porté vers la définition d'une technique qui s'appuie sur le principe de la Programmation par Démonstration afin de grandement faciliter et accélérer la description des différentes activités proposées par la leçon (4.3.1.1). Dans la même optique, le formateur aura la possibilité de piloter l'EF et ainsi de pouvoir le placer dans un contexte spécifique correspondant à un instant particulier du déroulement de la leçon. Cela se révèlera particulièrement utile dans certaines situations. Enfin, nous préconisons une méthode permettant de s'assurer de la cohérence globale du scénario en vérifiant que l'ensemble des activités décrites est réalisable dans le cadre des deux formes de scénario qu'il est possible de créer (4.3.1.2).

4.3.1.1 Génération automatisée de l'Activité

Afin de décrire le déroulement de la leçon de manière plus rapide et plus sûre, nous avons imaginé un processus permettant d'automatiser cette opération (Figure 25). Ce processus repose sur la technique de Programmation par Démonstration (voir 2.3.1.2), dans laquelle l'idée principale est de demander à une personne de réaliser une séquence d'actions dans un environnement informatique qui est alors enregistrée dans un format particulier. Il est ensuite possible de réutiliser cette séquence afin par exemple de rejouer les actions associées dans l'environnement informatique correspondant, ou de la comparer avec les actions effectuées par une autre personne pour indiquer d'éventuelles divergences.

Notre Environnement de Formation permettant de surveiller les changements d'état de ses différents modules au travers du Contexte Partagé (voir partie 3.2.2.4), il nous est possible d'appliquer cette technique dans la méthodologie que nous développons dans ce chapitre pour permettre à des formateurs de décrire des scénarios sans compétence informatique particulière. Ainsi, l'environnement auteur permettra au formateur d'exécuter l'EF et d'interagir avec ses différents modules. En étant notifié des changements d'état associés, il sera alors en mesure d'instancier automatiquement des séquences de bloc d'activité associés les valeurs adéquates pour chaque propriété. Les blocs ainsi instanciés ont le type :

- *Checkpoint* par défaut (ex : interaction dans l'Environnement Virtuel ou dans un Logiciel Métier, …).
- Animation s'il n'y a pas de bloc de type *Checkpoint* correspondant à la propriété qui vient d'être modifiée (ex: changement d'environnement et de point de vue dans l'Environnement Virtuel, ou activation/désactivation d'un comportement dans les systèmes de simulation).

Bien sûr, le formateur aura toujours la possibilité par la suite de modifier la représentation graphique ainsi générée, en ajoutant des blocs manuellement afin par exemple de créer des activités alternatives ou parallèles.

Grâce à cette méthode, le formateur est en mesure de décrire rapidement la dimension centrale du scénario, c'est-à-dire le déroulement des activités. Il peut alors ensuite se concentrer sur la description des dimensions pédagogiques et ludiques qui sont à sa portée et constituent des

connaissances à transmettre qu'il nous semble important de personnaliser pour être parfaitement adaptées à la leçon étudiée.

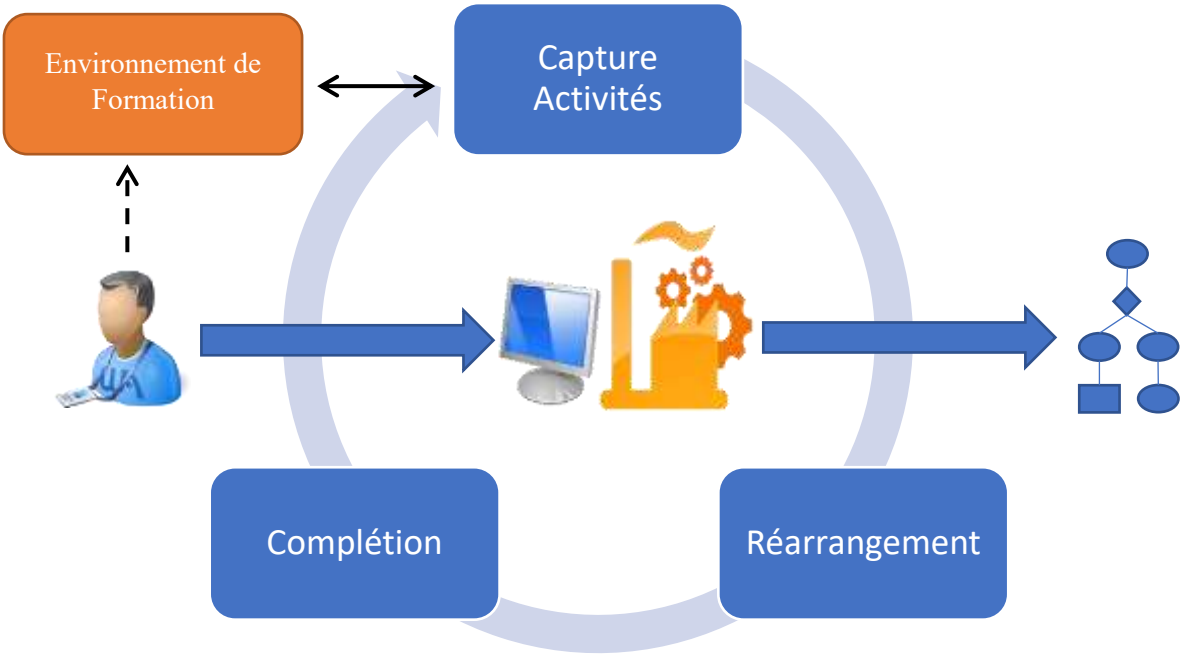


FIGURE 25. AUTOMATISATION DU PROCESSUS DE DESCRIPTION DE L'ACTIVITE

4.3.1.2 Assurer la cohérence de la description de l'Activité

Bien que la méthode d'automatisation de description de l'Activité permette d'accélérer le processus et de réduire le nombre d'erreurs liées à l'identification des propriétés et valeurs souhaitées pour faire progresser le scénario, elle ne permet pas de prévenir les erreurs de "logique" dans l'enchainement des opérations. Cela peut arriver dans différentes circonstances, par exemple (de manière non-exhaustive) :

- l'auteur décrit l'Activité entièrement manuellement ;
- l'auteur lie entre elles deux séquences de blocs générés automatiquement ;
- l'auteur ajoute des blocs manuellement au milieu d'une séquence ;
- l'auteur crée des activités parallèles ou alternatives à partir de deux séquences existantes (décrites manuellement ou générées automatiquement).

Dans ces cas-là, même si la représentation graphique de l'Activité est correcte d'un point de vue syntaxique, il *peut* arriver que des incohérences apparaissent à l'exécution du scénario. En particulier dans le cas des scénarios entièrement guidés, ces incohérences pourraient prendre la forme d'une action demandée à l'apprenant qui n'est pas disponible dans le contexte actuel (ex : dans le logiciel de diagnostic, cliquer sur un bouton qui n'existe pas sur l'écran courant). Cela pourrait alors entrainer l'affichage de messages pédagogiques d'erreur inopportuns, voire même un blocage du scénario où le joueur se retrouve dans une situation où il ne peut plus progresser. En plus d'une erreur effectuée par l'auteur lors de la description manuelle de l'Activité, ces incohérences peuvent aussi apparaitre lors de l'utilisation d'activités parallèles. La nature même de ces activités implique que plusieurs d'entre elles doivent être effectuées afin de pouvoir réaliser la suite des opérations ; à la fin d'une des activités concernées, l'utilisateur doit donc en

commencer une autre en effectuant une action particulière qui correspond au bloc de Checkpoint qui débute la branche souhaitée. Cependant, l'état de l'EF ayant été modifié lors de l'exécution de la branche précédente, l'action en question en question peut être devenue inaccessible, provoquant ainsi le même problème qu'évoqué plus haut.

Afin de répondre à cette problématique, nous enrichissons notre méthodologie visant à rendre accessible la description des scénarios aux formateurs avec deux nouvelles fonctionnalités. La première fonctionnalité concerne l'intégration dans notre environnement auteur d'un processus permettant de calculer puis restaurer l'état global de notre Environnement de Formation à toute étape de la description du scénario. Cela permet à l'auteur de se replacer dans un contexte particulier afin de s'assurer que les opérations qui suivent sont disponibles et n'entraineront pas d'incohérence à l'exécution de la leçon. Pour cela :

1. l'environnement auteur doit être capable de calculer dynamiquement l'état global de l'EF associé à chaque bloc d'activité dans la description du déroulement de la leçon. Cette opération est possible car chaque bloc étant intrinsèquement associé à un ou plusieurs changements de propriétés de l'EF, il est possible de déduire l'état du système à chaque étape de la description de l'Activité en partant de l'état initial de l'EF. Nous observons deux cas particuliers qui peuvent s'entrecroiser :
    - Si le bloc étudié se trouve à la suite de plusieurs structures de contrôle, l'environnement auteur doit demander à l'utilisateur quelles sont les branches qui ont été effectuées dans le cas d'activités alternatives. En ce qui concerne les activités parallèles, l'environnement auteur doit demander à l'utilisateur dans quel ordre elles ont été effectuées. Ces indications sont en effet nécessaires car l'état de l'EF peut être différent en fonction des activités ayant été effectuées et de l'ordre dans lequel elles ont été effectuées.
    - Si le bloc étudié se trouve dans une structure de contrôle d'activités parallèles, l'environnement auteur doit demander à l'utilisateur de spécifier les branches qui ont déjà été effectuées (et qui ont donc modifié l'état de l'EF). Dans le cas d'activités alternatives, leur nature même fait qu'il n'est pas nécessaire de considérer les branches autres que celle qui contient le bloc, car elles ne seront alors pas effectuées.
2. l'environnement auteur doit être en mesure de piloter l'Environnement de Formation afin de restaurer l'état global associé à un bloc d'activité particulier et qui aura été calculé précédemment. Cela est rendu possible grâce au Contexte partagé mis à disposition par l'EF et qui permet d'effectuer des requêtes de manière uniforme sur les différents modules interfacés afin par exemple d'en modifier l'état. Ces requêtes débouchent ensuite sur des résultats concrets en fonction du module :
    - pour l'Environnement Virtuel, le modèle présenté dans la partie 3.2.2.1 autorise les changements d'état de chaque propriété sans contrainte particulière ; les requêtes effectuées déboucheront donc sur une modification effective et directe de l'état du module.
    - en ce qui concerne les Logiciels Métier (3.2.2.2) et les Systèmes de Simulation (3.2.2.3), les modèles décris dans les parties respectives étant des modèles dégradés ne représentant pas nécessairement l'ensemble des systèmes associés de manière

exhaustive, il n'est pas possible d'assurer leur passage d'un état A vers un état B directement. Dans ce cas précis, deux solutions sont possibles :

- si les applications associées proposent une interface permettant de simuler des interactions, il est possible de rejouer le scénario de manière autonome jusqu'au bloc étudié (en respectant les branches effectuées et l'ordre dans lequel elles ont été effectuées dans le cas des activités alternatives et parallèles). Seul les blocs concernant des propriétés liées aux LM ou aux SS seront alors traités, en simulant donc uniquement les interactions devant être effectuées par le joueur et qui sont décrites au travers des blocs de type Checkpoint.
- dans le cas contraire, le processus sera identique excepté que l'environnement auteur demandera à l'utilisateur d'effectuer les interactions associées aux blocs qu'il ne peut pas exécuter de manière autonome. Si l'action effectuée ne correspond pas au changement d'état attendu, une information sera apportée à l'auteur. Celui-ci aura alors la possibilité d'effectuer une autre action s'il estime que celle qu'il vient d'effectuer n'a pas d'influence sur la suite du scénario, ou il pourra repartir de l'état initial (et donc recommencer l'opération depuis le début du scénario).

La deuxième fonctionnalité concerne l'intégration dans notre environnement auteur d'un deuxième processus qui se déroule en aval de la description du scénario et qui permet de vérifier sa cohérence générale. L'objectif ici est de fournir à l'utilisateur un moyen final de vérifier que l'ensemble du scénario qu'il a décrit est jouable et cohérent, indépendamment des activités alternatives choisies et de l'ordre dans lesquelles les activités parallèles auront été effectuées. La première étape de ce processus sera donc de générer l'ensemble des combinaisons possibles dans le choix des activités à effectuer. La deuxième étape sera alors de vérifier que chaque combinaison est jouable et cohérente. Tout comme pour le pilotage de l'Environnement de Formation explicité plus haut, cette deuxième étape est contrainte par les actions qu'il est possible d'effectuer directement ou non sur les modules des Logiciels Métier et des Systèmes de Simulation :

- s'ils sont capables de simuler des interactions de manière autonome, alors la combinaison entière est jouée depuis le début de manière automatique par l'environnement auteur. Seuls les blocs de type *Checkpoint* concernant des interactions dans les différents modules de l'EF sont alors traduits en requêtes de modification sur les systèmes associés (les autres blocs sont traités normalement : ceux de type *Checkpoint* par une observation des propriétés et de leur valeur, et ceux de type *Animation* par une modification de valeur des propriétés concernées).
- s'ils n'en sont pas capables, l'environnement auteur sollicitera l'utilisateur afin qu'il valide chaque combinaison manuellement. Pour effectuer cette validation correctement, chaque interaction effectuée par l'auteur dans l'Environnement de Formation devra être en correspondance avec le bloc de type *Checkpoint* courant. Dans le cas contraire, une alerte sera levée afin de prévenir l'utilisateur qu'il serait préférable qu'il recommence la validation depuis le début du fait qu'il a effectué une interaction non attendue.

Finalement, les deux processus décris dans cette partie permettent d'assurer la cohérence globale du scénario et plus particulièrement celle de la description de l'Activité en deux étapes complémentaires :

- calcul et restauration de l'état de l'Environnement de Formation associé aux blocs d'activités afin de prévenir les erreurs en cours de description.
- validation finale du scénario pour s'assurer que l'ensemble des combinaisons induites par l'utilisation d'activités alternatives et parallèles sont jouables.

Cela est bien entendu utile afin de valider la cohérence des scénarios lors de leur description initiale, mais cela peut aussi être très utile dans le cadre d'évolutions apportées aux différents modules de l'Environnement de Formation (lors de mises à jour par exemple) afin de vérifier que les scénarios créés précédemment sont toujours jouables.

4.3.2 Promouvoir la réutilisabilité

Un deuxième grand principe que nous incluons dans notre méthodologie afin d'améliorer le processus global de description des scénarios concerne la capitalisation de différents éléments ayant déjà été modélisés dans le cadre de scénarios précédents. En proposant plusieurs catégories de *bibliothèques*, l'environnement auteur permettra alors à l'utilisateur de réutiliser des séquences de blocs d'activités (4.3.2.1), des retours pédagogiques ou encore des éléments de mise en scène (4.3.2.2) pour décrire de nouveaux scénarios. L'objectif est double : éviter de demander à l'auteur de modéliser plusieurs fois un même élément, tout en accélérant le processus de description des différentes dimensions de nos scénarios.

4.3.2.1 Blocs composites

Les premiers éléments que nous avons souhaité pouvoir capitaliser sont les séquences de blocs d'activité susceptibles d'être communes à plusieurs scénarios. Par exemple, pour toutes les procédures de diagnostic de véhicule, une phase initiale d'établissement de connexion entre le logiciel de diagnostic et la voiture est requise. Il serait donc intéressant d'avoir la possibilité de décrire une fois pour toute cette séquence initiale, pour pouvoir ensuite la réutiliser directement dans tous les scénarios qui en ont besoin.

Dans un premier temps, il est nécessaire de pouvoir créer ces séquences réutilisables. Pour cela, nous nous appuyons sur la notion de sous-activité introduite dans la partie 4.2.1.4. A partir de sous-activités décrites dans la modélisation du déroulement de la leçon, l'environnement auteur permettra alors de créer de nouveaux blocs d'activité, que nous qualifions de *composite*, qui encapsulent les sous-activités associées et qui sont donc définis par :

- Une ou plusieurs instances de blocs d'activité qui en marquent le début.
- Une ou plusieurs instances de blocs d'activité qui en marquent la fin.
- Les instances de bloc d'activité qui en décrivent le comportement (c'est-à-dire l'évolution de l'Environnement de Formation entre les blocs de début et les blocs de fin).
- Les transitions qui décrivent la ou les séquences de blocs encapsulés.
- Une liste de propriétés configurables. Ces propriétés doivent faire référence à des propriétés associées aux instances de bloc utilisées pour décrire ce nouveau bloc. De la même manière que pour les blocs d'activités de base, cela permet de proposer une certaine flexibilité dans la création et l'utilisation de ces blocs composites, afin de pouvoir les utiliser dans différents contextes. Par exemple, dans le cas de la phase initiale d'établissement de communication entre le logiciel de diagnostic et le véhicule, la

sélection manuelle du véhicule se fait en cliquant sur le bouton associé au modèle du véhicule à diagnostiquer. En définissant une propriété configurable qui fait référence à la propriété identifiant le bouton devant être cliqué pour la sélection manuelle du véhicule, il est possible de créer un bloc composite utilisable quel que soit le modèle de la voiture. Il suffira alors seulement de spécifier la valeur souhaitée pour cette propriété lors de l'instanciation du bloc composite.

Dans un second temps, ces blocs composites viennent compléter la liste des blocs disponibles pour décrire l'Activité, afin de pouvoir être réutilisés dans la description du scénario actuel ou de futurs scénarios. La Figure 26 présente les modifications apportées au modèle de la description de l'Activité donné en 4.2.1 afin de prendre en compte ce nouveau type de bloc :

- Les blocs d'activité disponibles pour la description du déroulement de la leçon font désormais la distinction entre les blocs "de base" (ceux que l'on considérait jusque maintenant) et les blocs composites, qui eux peuvent donc encapsuler ces deux formes de blocs. De manière similaire, les blocs d'instance font aussi cette distinction.
- Les blocs d'activité peuvent maintenant être de type `Composite`. Graphiquement, les instances de ces blocs composites sont représentées par une forme rectangulaire aux coins arrondis et de couleur blanche (quelle que soit la catégorie à laquelle ils sont associés).
- En plus d'une référence vers l'instance de bloc à laquelle elles sont liées graphiquement, les transitions doivent maintenant expliciter en plus les instances de blocs "de base" source et cible auxquelles elles sont liées. En effet, les blocs composites en eux-mêmes ne sont qu'une manière de représenter graphiquement et de manière allégée un ensemble de séquences de blocs "de base". Les transitions vers et depuis ces blocs composites doivent donc connaître les instances de bloc "de base" effectives auxquelles elles sont liées afin de décrire correctement les séquences d'opérations de l'Activité. Graphiquement, ces transitions devront donc donner le label de l'instance de bloc depuis laquelle elles partent et/ou qu'elles ciblent aux extrémités correspondantes.

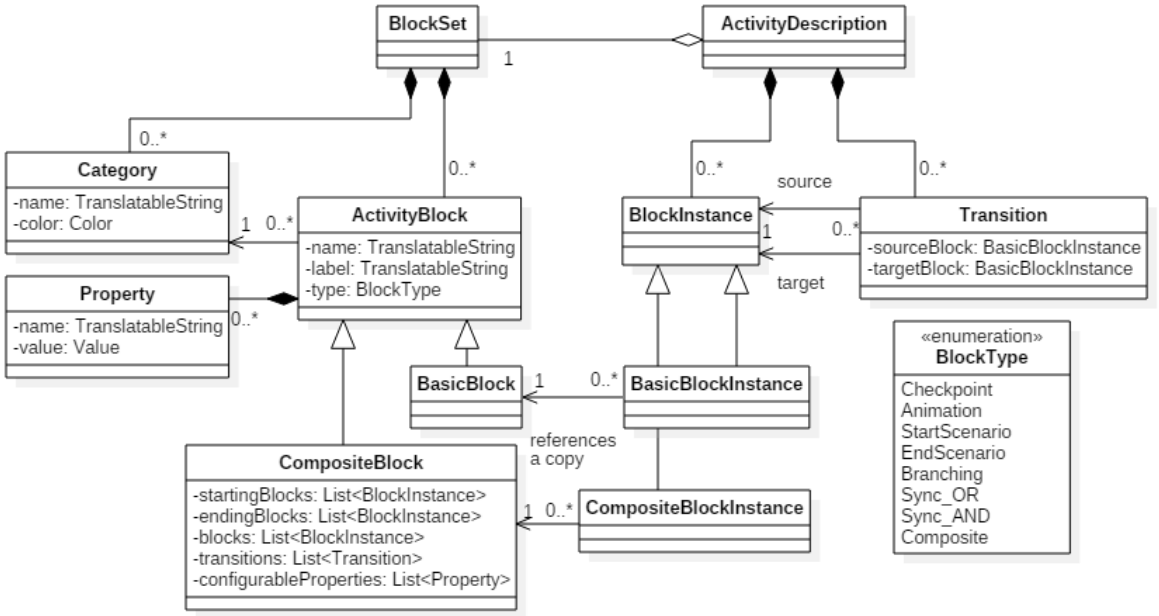


FIGURE 26. MISE A JOUR DU MODELE DE DESCRIPTION GRAPHIQUE DE L'ACTIVITE (LES SOUS-ACTIVITES AINSI QUE LES ETIQUETTES PEDAGOGIQUES ET LUDIQUES SONT VOLONTAIREMENT OMISES POUR NE PAS SURCHARGER LE DIAGRAMME)

L'environnement auteur devra par ailleurs permettre d'accéder au contenu des blocs composites (instanciés ou non) afin de les examiner et éventuellement de les modifier à n'importe quel moment (localement ou globalement).

Finalement, la méthode décrite dans cette partie permet de capitaliser des séquences de blocs d'activité en fournissant à l'auteur un moyen :

- de créer ses propres blocs composites à partir de séquences d'instances de bloc de base et ainsi de compléter la liste des blocs d'activités.
- de réutiliser ces blocs composites dans tous les scénarios qu'il décrira par la suite.

4.3.2.2 Retours pédagogiques et éléments de mise en scène

La deuxième grande famille d'éléments qu'il nous a semblé important de capitaliser concerne les différentes instances d'éléments pédagogiques et ludiques prises en compte par notre Environnement de Formation et définies spécifiquement pour chaque scénario. En effet, les dimensions pédagogiques et ludiques étant des aspects du scénario devant être définis entièrement manuellement par le formateur dans le cadre de notre méthodologie, elles sont de nature à requérir un travail pouvant parfois être long et fastidieux. Ainsi, nous proposons dans le cadre de ces travaux un processus qui permettra à l'auteur de réutiliser des instances d'éléments pédagogiques et ludiques définis dans des scénarios précédents.

Pour cela, l'environnement auteur mettra à disposition de l'utilisateur deux grandes fonctionnalités permettant respectivement :

- de sélectionner un ensemble de retours pédagogiques et d'éléments de mise en scène associés au scénario courant pour les exporter vers une base de données générale. Ces éléments seront ensuite disponibles globalement lors de la description de n'importe quel scénario.
- d'importer des retours pédagogiques et des éléments de mise en scène à partir d'un scénario précédent. Ce scénario sera ensuite analysé afin d'abord de présenter à l'utilisateur l'ensemble des éléments associés ayant été décrits, puis de lui permettre de choisir ceux qu'il souhaite importer. L'importation pourra se faire :
    - localement, c'est-à-dire que les éléments importés seront directement associés au scénario courant ;
    - globalement, c'est-à-dire que les éléments importés seront copiés vers une base de données générale disponible lors de la création de n'importe quel scénario.

Ces deux fonctionnalités sont complémentaires et sont disponibles afin de permettre à l'auteur de s'en servir lorsqu'il en a besoin ou lorsqu'il estime que cela est pertinent, sans lui imposer de contrainte supplémentaire lorsqu'il travaille sur la description de ses scénarios. La base de données regroupant tous les éléments pédagogiques et ludiques globaux à l'ensemble des scénarios devra être consultable et modifiable à tout instant. De plus, l'utilisation d'éléments de cette base de données pour la description de scénarios devra créer une copie locale des éléments en question afin qu'ils puissent être éventuellement modifiés et adaptés au contexte spécifique de la leçon en cours de modélisation.

Finalement, la mise en place d'une base de données associée à des fonctionnalités d'importation et d'exportation dans notre environnement auteur permet de faciliter et d'accélérer la description des dimensions pédagogiques et ludique en permettant à l'utilisateur :

- d'éviter de devoir spécifier plusieurs fois un même élément qu'il sait qu'il aura à utiliser dans plusieurs scénarios (ex : message d'aide "appuyez sur le bouton de Validation dans le logiciel de diagnostic").
- de pouvoir utiliser des éléments existants en tant que "squelettes" qui seront modifiables pour être adaptés au contexte courant du scénario actuel ou de l'activité actuelle. Cela se révèle utile pour les éléments un plus complexes ou plus longs à spécifier (ex : dialogues, questionnaires, etc.).

4.4 Synthèse

Dans ce chapitre, nous nous sommes intéressés à présenter dans le cadre de notre outil de formation une méthodologie visant à rendre la création de scénarios accessible à des formateurs qui ne possèdent pas nécessairement de compétences informatiques particulières. Cette méthodologie est centrée sur l'utilisation d'un environnement auteur qui propose plusieurs fonctionnalités permettant d'atteindre ce but.

Dans un premier temps, nous avons introduit un langage graphique, associé au langage textuel de description de scénario introduit dans la partie précédente, qui sera manipulable au travers de notre environnement auteur et qui propose de :

- représenter séparément et dans un formalisme graphique à base de blocs et de transitions les trois grandes dimensions de nos scénarios (Activité, Pédagogie, éléments ludiques) afin de s'adapter à l'expertise du formateur en ne mélangeant pas des concepts différents par nature.
- faire l'association entre ces trois dimensions en se reposant sur la notion de sous-activité qui permet de lier des séquences d'opérations importante de la description de l'Activité aux activités pédagogiques de la dimension pédagogique et aux missions de la dimension ludique.

Une fois le scénario décrit à l'aide de ce langage graphique, l'environnement auteur permettra de l'exporter automatiquement vers sa représentation dans le langage dédié. Pour cela, notre méthodologie présente un ensemble de règles permettant de valider la syntaxe de la représentation graphique du scénario, ainsi que le processus permettant de transposer les différents éléments graphiques vers les instructions correspondantes de notre langage dédié.

Dans un second temps, nous avons présenté un ensemble de fonctionnalités permettant d'améliorer le processus global de description des scénarios et ainsi d'alléger la charge de travail du formateur :

- utilisation de l'Environnement de Formation afin de générer automatiquement des séquences d'activité et d'assurer la cohérence finale de la description de l'Activité.
- mise en place d'un système de capitalisation permettant de réutiliser dans des scénarios futurs différents éléments décrits dans le cadre de scénarios précédents.

Partie III. Réalisations et Évaluations

Chapitre 5. Application au contexte industriel diagnostic automobile

5.1 Introduction

Nous mettons dans ce chapitre l'accent sur les différents environnements ayant été développés tout au long de cette thèse afin d'appliquer nos différentes contributions au contexte industriel du diagnostic automobile auprès d'un des clients d'ACTIA : le groupe PSA. Ainsi, nous présentons dans un premier temps l'outil de formation que nous avons implémenté afin de mettre en application les différents concepts abordés dans le 0 (5.2). En particulier, nous décrivons les différents modules qui composent notre Environnement de Formation, ainsi que le moteur d'exécution des scénarios permettant aux apprenants de suivre les leçons modélisées. Enfin, nous introduisons le projet Diag'Adventures qui nous a permis d'évaluer la pertinence de l'outil de formation proposé.

Dans un second temps, nous nous intéressons à l'environnement auteur que nous avons développé afin de mettre en pratique la méthodologie de création des scénarios décrite dans le Chapitre 4 (5.3). Nous décrivons alors les différentes fonctionnalités implémentées et les interfaces utilisateurs correspondantes, ainsi que de nouveaux éléments nous permettant d'améliorer le processus de création des scénarios. Nous finissons par expliquer les difficultés rencontrées pour évaluer cet environnement auteur et proposons une alternative afin d'apprécier les avantages et les inconvénients de l'outil implémenté.

5.2 Outil de Formation

Comme nous l'avons présenté dans le 0, l'outil de formation proposé s'appuie sur une architecture modulaire afin de pouvoir mêler dans une même application l'utilisation d'un Environnement Virtuel, de Logiciels Métier, et de Systèmes de Simulation. Afin de mettre en pratique nos contributions, nous avons implémenté un outil de formation qui instancie ces différents modules et qui permet ainsi d'enseigner des activités se plaçant dans le contexte industriel du diagnostic automobile (Figure 27).

Dans cette partie, nous présentons donc dans un premier temps l'Environnement de Formation que nous avons implémenté afin de reproduire de telles activités dans l'environnement de travail quotidien du mécanicien (5.2.1). En plus de l'Environnement Virtuel qui reproduit l'ensemble des objets physiques impliqués (garage, véhicule, tournevis, etc.), notre EF intègre alors un logiciel de Diagnostic (DiagBox) permettant d'effectuer diverses opérations sur le véhicule, ainsi qu'une application permettant de simuler les trames échangées entre le véhicule diagnostiqué et DiagBox. Puis, nous explicitons le moteur d'exécution que nous avons implémenté afin d'organiser l'ensemble de ces modules, d'exécuter les scénarios associés, et de présenter les différents éléments pédagogiques et ludiques requis (5.2.2). Enfin, nous présentons les tests terrains ayant été conduits afin d'évaluer la pertinence de notre approche (5.2.3).

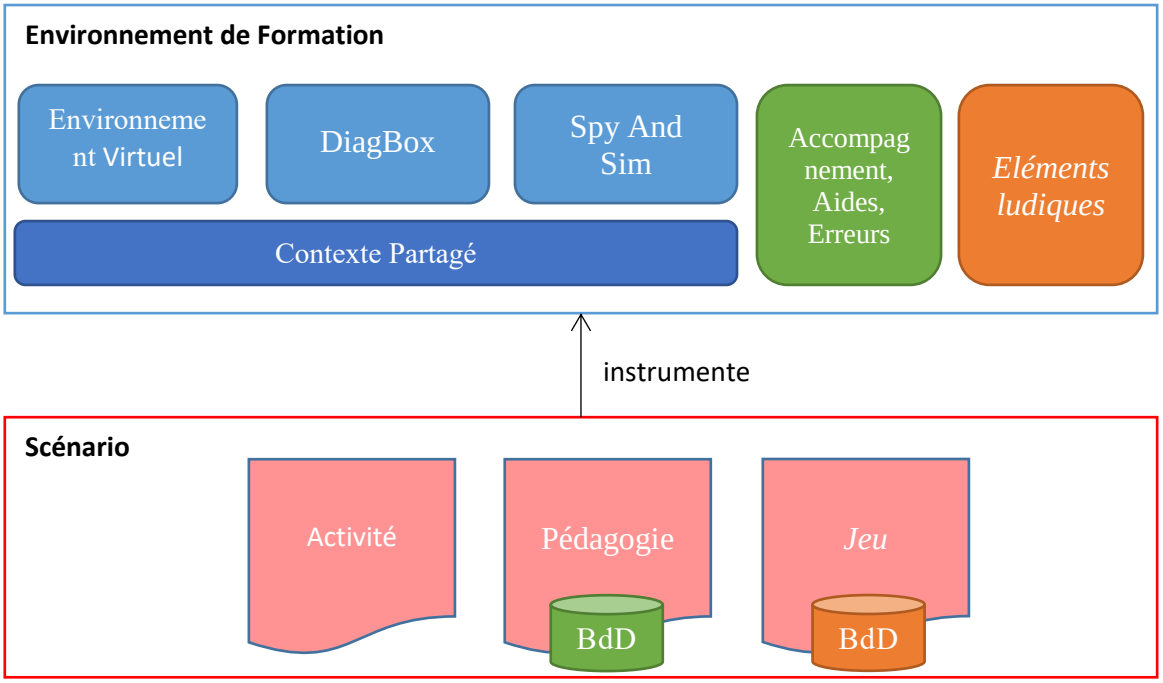


FIGURE 27. OUTIL DE FORMATION IMPLEMENTE POUR L'ENSEIGNEMENT D'ACTIVITES SE PLAÇANT DANS LE CONTEXTE INDUSTRIEL DU DIAGNOSTIC AUTOMOBILE.

5.2.1 Environnement de Formation

Dans cette partie, nous nous intéressons aux différents modules qui composent l'Environnement de Formation que nous avons implémenté afin de l'appliquer à l'enseignement d'activités métier dans le contexte industriel du diagnostic automobile. Nous présentons ainsi l'implémentation de l'Environnement Virtuel et de ses différentes fonctionnalités en 5.2.1.1, le logiciel de diagnostic automobile et les développements effectués afin de pouvoir l'interfacer avec des systèmes externes en 5.2.1.2, et l'outil de simulation des trames véhicules en 5.2.1.3. Enfin, nous décrivons en 5.2.1.4 le contexte partagé permettant d'interfacer ces trois modules et de centraliser l'ensemble des informations requises pour décrire l'état général de l'Environnement de Formation.

5.2.1.1 L'Environnement Virtuel

Afin de reproduire les différents environnements susceptibles d'intervenir dans les scénarios de formation étudiés, nous avons conçu et implémenté à l'aide du moteur de jeu Unity [28] un Environnement Virtuel qui instancie le modèle décrit dans la partie 3.2.2.1. Une de nos intentions lors du développement de cette application était de la rendre modulable, c'est-à-dire qu'elle puisse être personnalisée en fonction des besoins des scénarios futurs sans avoir à la recompiler. Comme nous allons le voir dans cette partie, cet objectif initial n'a été rempli que partiellement car nous avions besoin de parvenir à un prototype fonctionnel rapidement, afin d'être en mesure de mettre en application les travaux présentés dans le 0 et le Chapitre 4.

[28] https://unity3d.com

La Figure 28 présente une capture d'écran de l'Environnement Virtuel utilisé dans le cadre de l'exemple donné dans la partie 3.3.2. Dans un souci de simplicité, nous avons ici opté pour un rendu visuel purement 2D, mais nous verrons dans la partie 5.2.3 que l'utilisation d'un rendu visuel 3D amène à un résultat similaire. Les images utilisées dans notre EV en 2D sont d'ailleurs directement issues cet EV en 3D. La principale différence réside dans la gestion des points de vue :

- en 3D, il est possible de modifier les propriétés de la caméra (position, angle, ouverture, etc.) librement, ou de prédéfinir des valeurs particulières pour ces propriétés (afin de définir l'ensemble des Points de Vue disponibles dans chaque environnement). Dans les deux cas, c'est ensuite le rôle du moteur graphique de se charger de définir comment les différents objets de l'environnement sont perçus par l'utilisateur.
- en 2D, si l'utilisateur souhaite percevoir la scène sous un autre angle (afin par exemple d'observer un élément plus précisément), il faut que le développeur ait prévu le point de vue au préalable, c'est-à-dire les images associées aux différents éléments visibles et leur position à l'écran. Bien que cette solution demande plus de travail (plus il y a de points de vue dans un environnement, plus il faut d'images afin de représenter un même élément), nous avons choisi de l'utiliser car elle restait plus simple et rapide à mettre en place qu'une solution requérant la création de modèles 3D.

FIGURE 28. CAPTURE D'ECRAN DE L'ENVIRONNEMENT VIRTUEL IMPLEMENTE. IL REPRESENTE UN GARAGE ET LE VEHICULE AVEC LEQUEL DEVRA INTERAGIR L'APPRENANT

Ainsi, l'utilisateur peut naviguer dans notre Environnement Virtuel 2D et accéder aux différents objets présents dans notre modèle à l'aide de deux menus déroulants : le premier permet

de choisir l'environnement (garage, route, etc.), alors que le deuxième permet de choisir le point de vue souhaité pour l'environnement courant. Ensuite, si la souris se trouve par-dessus un élément interactif, celui-ci se met en surbrillance. En cliquant, l'utilisateur fait alors apparaitre un nouveau menu déroulant qui présente l'ensemble des interactions associées à cet élément. Il ne lui reste alors plus qu'à cliquer sur l'interaction souhaitée afin de l'effectuer, entrainant alors les changements d'états qui lui sont attachés.

Afin de prendre en compte la gestion des points de vue dans un espace 2D d'un côté, et afin de rendre cet espace le plus modulable possible de l'autre :

- nous avons complété le modèle proposé dans la partie 3.2.2.1 tel que montré sur la Figure 29. Les éléments suivants ont ainsi été ajoutés (mais ils ne modifient pas les propriétés du modèle initial nécessaires pour représenter l'état global de l'EV) :
    - ***Décor***. Chaque environnement possède maintenant un décor, qui représente l'ensemble des objets statiques du décor. Lorsqu'un environnement est visible, son décor l'est donc aussi (il n'est pas possible de le cacher au contraire des différents éléments qui composent l'environnement).
    - ***Instance du décor***. Elle représente la manière dont est perçu le décor en fonction du point de vue actuel dans l'environnement courant. Chaque point de vue est donc maintenant associé à une instance du décor, cette instance étant liée à une image représentant la manière dont l'utilisateur perçoit le décor en fonction du point de vue sélectionné.
    - ***Instance d'élément***. Elle représente la manière dont est perçu chaque élément de l'environnement associé en fonction du point de vue actuel. Chaque point de vue est donc maintenant associé à une instance d'élément, qui elle est associée à une ou plusieurs images. Ces images représentent les différentes manières dont l'utilisateur perçoit l'élément en fonction du point de vue sélectionné, mais aussi en fonction de l'état de l'élément considéré. En effet, il est possible de vouloir représenter graphiquement un élément différemment en fonction de son état ou même de l'état général de l'Environnement Virtuel. Par exemple, l'élément "feu de croisement" possède l'état "est allumé". En fonction de la valeur associée à cet état (vrai ou faux), l'élément peut se représenter soit en étant allumé, soit en étant éteint.
- nous avons implémenté un système de chargement dynamique de notre modèle et des objets graphiques associés. Sans entrer dans le détail du fonctionnement d'Unity, l'utilisation classique de ce logiciel est de définir directement dans l'espace 2D/3D les objets visuels qui le composent ainsi que leurs différentes propriétés (position, comportement, etc.), avant de compiler le tout pour former l'exécutable final. Dans notre solution, afin de rendre notre Environnement Virtuel modulable, nous avons choisi de mettre en place un système qui vient charger et instancier dynamiquement ces différents objets à partir de deux fichiers XML qui décrivent le modèle souhaité de l'EV. De base, notre espace 2D ne contient donc aucun objet visuel lorsqu'il est compilé. Nous donnons en Annexe 4 un exemple des deux fichiers qui sont chargés pour effectuer cette instanciation dynamique.

Comme nous l'évoquions au début de cette partie, notre EV n'est cependant pas entièrement modulaire. En effet, la description de la position de chaque élément, de l'image utilisée pour les représenter à l'écran, ou plus généralement de leur comportement global en fonction de l'état de

l'EV, est effectuée directement dans Unity "en dur". En effet, il n'était pas aussi intuitif de décrire des comportements parfois complexes à l'aide de formats "standards" tels que JSON ou XML, et nous avons préféré nous concentrer sur l'implémentation d'une solution fonctionnelle vis-à-vis des délais imposés par cette thèse. Le développement d'un Environnement Virtuel entièrement modulable n'est cependant pas exclu et entre dans les Perspectives que nous présenterons à la fin de ce mémoire.

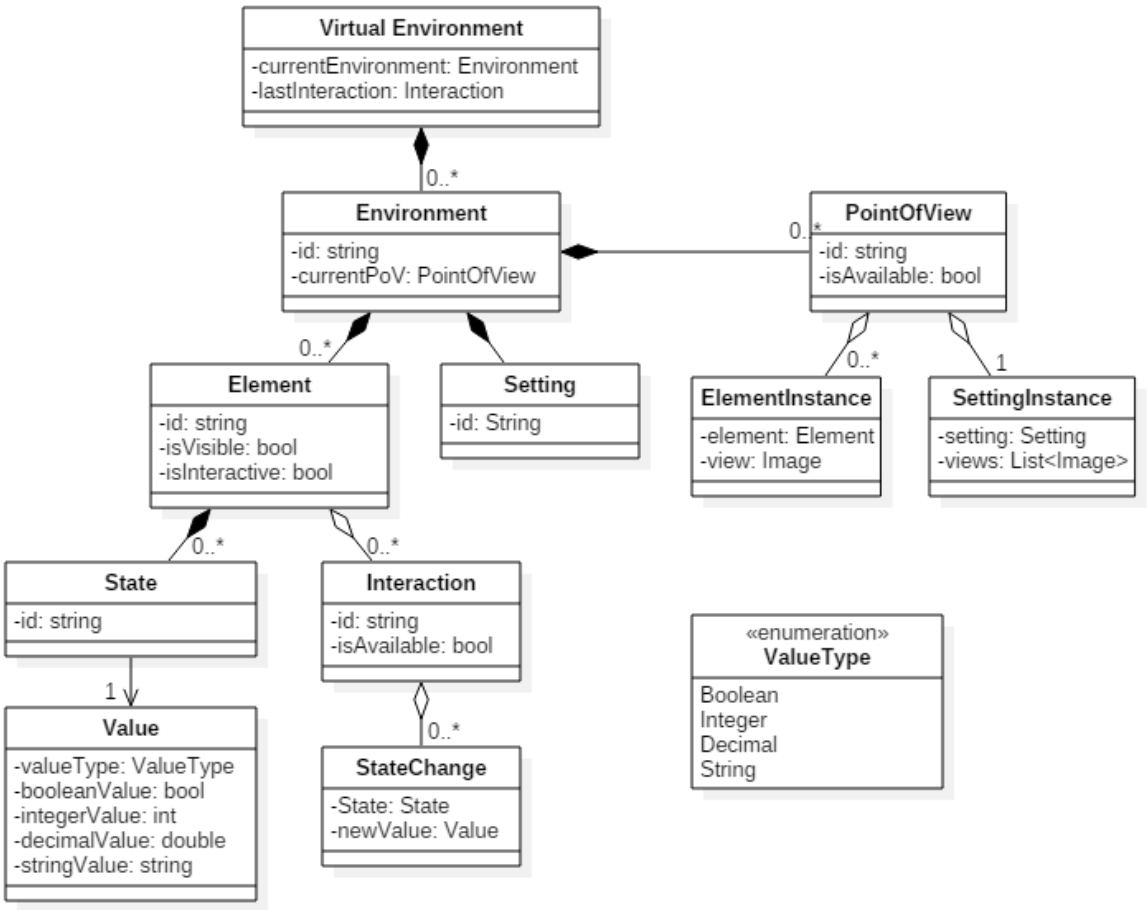


FIGURE 29. MODELE DE L'ENVIRONNEMENT VIRTUEL ENRICHI

Enfin, afin de pouvoir utiliser cet Environnement Virtuel dans le contexte plus global de l'Environnement de Formation présenté dans la partie 3.2, nous avons implémenté une interface permettant d'échanger des informations avec des applications extérieures via le protocole TCP/IP :

- via des notifications depuis l'EV vers les applications extérieures afin :
  - de signaler le changement de l'environnement courant ou du point de vue courant dans l'environnement actuel ;
  - de notifier la modification de différentes propriétés pour les objets de notre modèle (accessibilité des points de vue, visibilité et interactivité des éléments, valeur pour les différents états de chaque élément, et accessibilité des interactions) ;
  - de signaler / valider l'exécution d'une interaction. La fonction associée renvoie une valeur booléenne pour savoir si l'interaction doit effectivement être exécutée (voir partie 5.2.2.1) du côté de l'Environnement Virtuel.

- via des notifications depuis les applications extérieures vers l'EV afin :
    - de modifier l'environnement courant ou le point de vue courant dans l'environnement actuel ;
    - de modifier différentes propriétés des objets de notre modèle (accessibilité des points de vue, visibilité et interactivité des éléments, valeur pour les différents états de chaque élément, et accessibilité des interactions) ;
    - de simuler des interactions (c'est-à-dire de provoquer l'exécution d'une interaction sans intervention de la part de l'utilisateur) ;
    - ajouter de nouveaux éléments (plus de détails seront donnés dans la partie 5.3.3).

Finalement, la mise en place de cette interface permet au Contexte Partagé de maintenir à jour l'état global du module de l'Environnement Virtuel qui compose notre EF, et donc de mettre en application les contributions que nous avons décrites dans le 0 et le Chapitre 4 de ce mémoire.

5.2.1.2 Le logiciel de diagnostic : DiagBox

Les activités de diagnostic étudiées requièrent l'utilisation d'un logiciel de diagnostic qu'il faut connecter au véhicule afin d'échanger un certain nombre d'informations avec ce dernier. Un tel logiciel permet d'effectuer des opérations plus ou moins complexes sur le véhicule, allant de la simple analyse à la modification manuelle de certaines propriétés des composants électroniques (calculateurs) installés sur le véhicule. L'outil de formation que nous avons implémenté dans le cadre de notre étude utilise le logiciel DiagBox, c'est-à-dire l'outil de diagnostic spécialement développé par ACTIA pour le groupe PSA afin de diagnostiquer les véhicules de marque Peugeot et Citroën. Dans cette partie, nous cherchons à extraire dans un premier temps les différents éléments intéressants pour représenter l'état courant du logiciel DiagBox. Dans un second temps, nous présentons les différents développements effectués vis-à-vis de l'architecture du logiciel et qui permettent d'accéder à ces éléments depuis des environnements extérieurs.

La Figure 30 et la Figure 31 montrent deux exemples de capture d'écran du logiciel DiagBox. L'outil de diagnostic se présente globalement sous la forme d'écrans qui évoluent et/ou se succèdent dynamiquement en fonction des actions réalisées par l'utilisateur et/ou de l'*applicabilité véhicule*[29] diagnostiquée. De plus, DiagBox implémente la notion d'onglet, qui permet de présenter plusieurs écrans en même temps dans des vues différentes. Les onglets impactés par nos scénarios de formation sont au nombre de deux : l'onglet "Accueil" et l'onglet "Expert". Enfin, des pop-ups peuvent apparaitre au cours de la navigation dans l'outil, se superposant alors à l'onglet (et donc l'écran courant de cet onglet) auquel elle appartient.

Chaque écran ou pop-up est ensuite décrit par une hiérarchie d'éléments, parmi lesquels on va trouver ceux avec lesquels peut interagir l'utilisateur : les actions utilisateurs. Il existe plusieurs types d'action utilisateurs, telles que les cases à cocher ou les zones de texte, chacun étant défini par une ou plusieurs propriétés. Dans le cadre des scénarios de formation développés, l'unique

[29] Applicabilité véhicule : représente une "version particulière" d'un véhicule. En plus de la marque et du modèle, elle est entre autre identifiée par les calculateurs installés sur le véhicule, et les fonctionnalités accessibles. Cela représente par exemple les options qu'il est possible d'ajouter à l'achat d'un véhicule neuf.

type d'action utilisateur dont nous avons eu besoin est l'élément d'interface "Bouton", c'est-à-dire un élément qui permet d'effectuer une action lorsque l'utilisateur clique dessus.

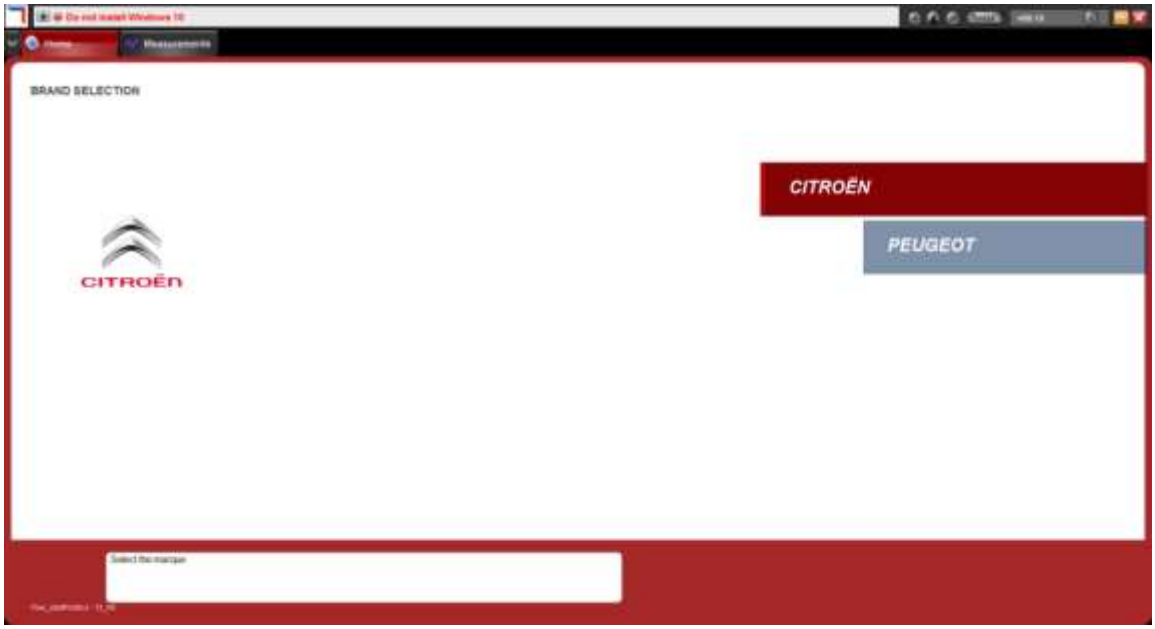


FIGURE 30. ECRAN D'ACCUEIL DE DIAGBOX : SELECTION DE LA MARQUE DU VEHICULE A DIAGNOSTIQUER (ONGLET ACCUEIL)

En nous référant à la partie 3.2.2.2, les éléments dont nous avons eu besoin afin de contrôler et de maintenir à jour l'état du système à chaque instant sont les suivants :

- Identifiant des écrans courants associés aux onglets respectifs "Accueil" et "Expert".
- Identifiant des pop-ups courantes associées aux onglets respectifs "Accueil" et "Expert".
- Liste des identifiants des actions utilisateur de type "bouton" disponibles pour chaque écran et pop-up courant.
- Identifiant de la dernière action utilisateur de type "bouton" effectuée, avec l'identifiant de l'onglet et de l'écran/pop-up associé.

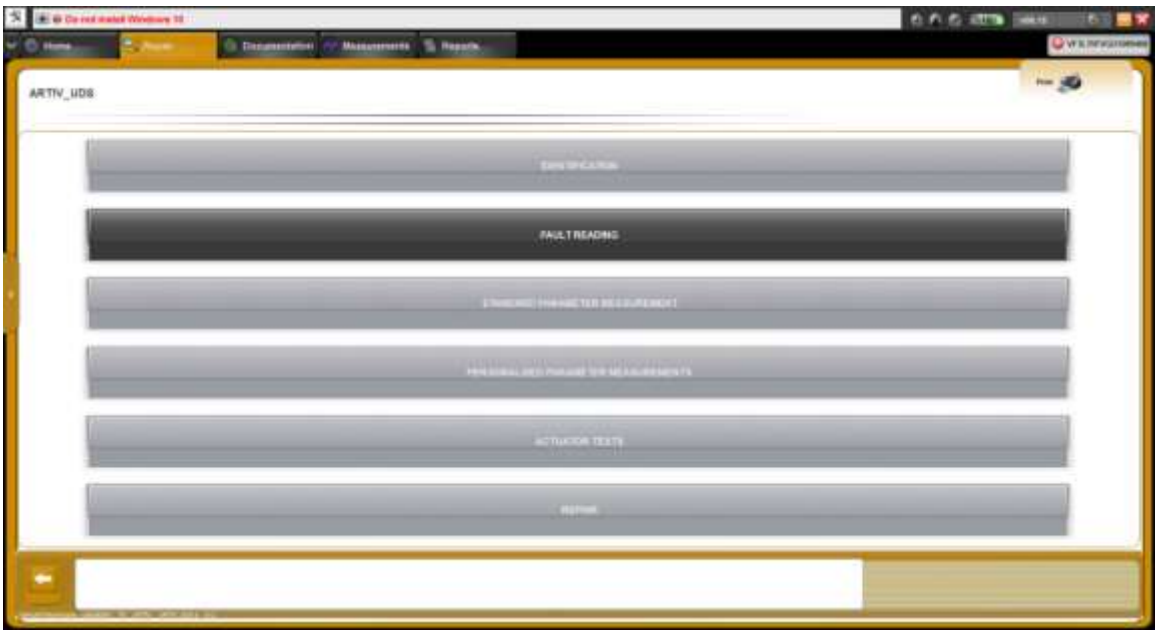


FIGURE 31. ECRAN AVANCE : OPERATIONS DISPONIBLES SUR LE CALCULATEUR DU RADAR ANTICOLLISION (ONGLET EXPERT)

La Figure 32 présente l'architecture n-tiers de DiagBox, avec ses principaux composants :

- l'Interface Homme-Machine. Elle représente la partie du logiciel directement visible et accessible par l'utilisateur. C'est donc via ce composant que l'utilisateur peut interagir et effectuer des actions utilisateurs.
- le module de communication. Il représente la partie du logiciel qui échange des trames avec le véhicule en cours de diagnostic pour accéder à certaines informations (en lecture ou en écriture) de ses différents composants électroniques.
- le moteur de diagnostic. Il représente la partie centrale du logiciel qui contrôle :
    - ce que doit afficher l'IHM en fonction des actions utilisateurs effectuées et des informations reçues via le module de communication.
    - les communications devant être effectuées avec le véhicule en fonction de la navigation dans l'outil.

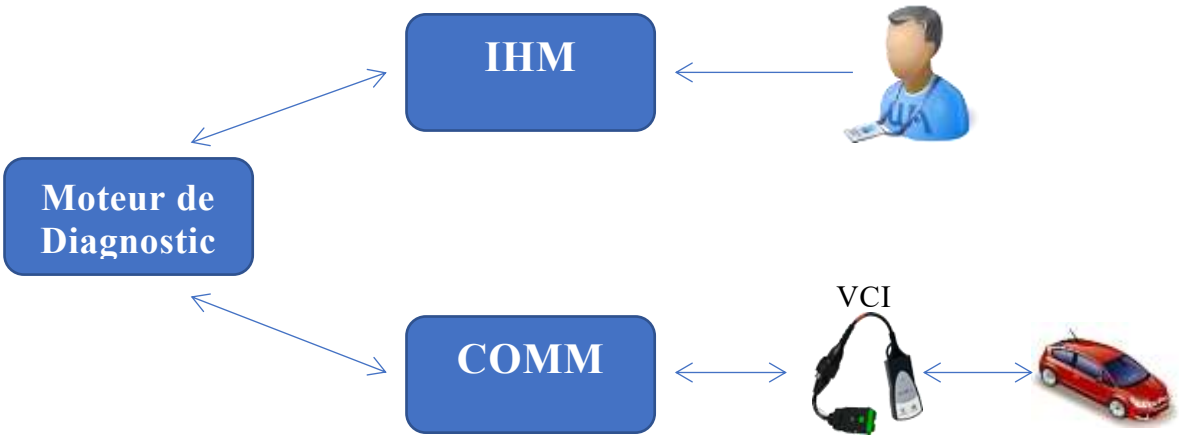


FIGURE 32. ARCHITECTURE N-TIER DE DIAGBOX (VISION SIMPLIFIEE)

En s'appuyant sur cette architecture, nous avons été en mesure d'implémenter et d'intégrer une application Homme-du-Milieu venant filtrer et traiter l'ensemble des échanges qui ont lieu entre le moteur de diagnostic et l'interface homme-machine (comme le montre la Figure 33). Cela nous a permis d'accéder aux informations intéressantes que nous avons présentées plus haut. Nous avons ensuite conçu et implémenté une interface afin que des applications extérieures à l'outil de diagnostic puissent :

- être notifiées de ces informations (construction d'un écran ou d'une popup, liste des actions utilisateur dans cet écran). En ce qui concerne la notification des actions utilisateur effectuées, la fonction associée renvoie une valeur booléenne afin de savoir si l'action utilisateur doit effectivement être exécutée du côté de DiagBox (voir partie 5.2.2).
- notifier des actions utilisateurs afin de piloter DiagBox manuellement sans intervention de la part d'un utilisateur.

Cette application tierce a été implémentée en utilisant le langage C++ et la librairie gSOAP[30] afin de s'interfacer avec les flux SOAP[31] échangés entre l'IHM et le moteur de diagnostic.

[30] https://www.genivia.com/dev.html

[31] https://www.w3.org/TR/soap/

L'interface extérieure mise à disposition utilise elle aussi la technologie SOAP et la librairie gSOAP afin de générer sa description en WSDL[32] et d'être ensuite utilisable depuis n'importe quelle application externe. Finalement, cette application Homme-du-Milieu nous a permis d'intégrer DiagBox dans notre Environnement de Formation en tant que module des Logiciels Métier.

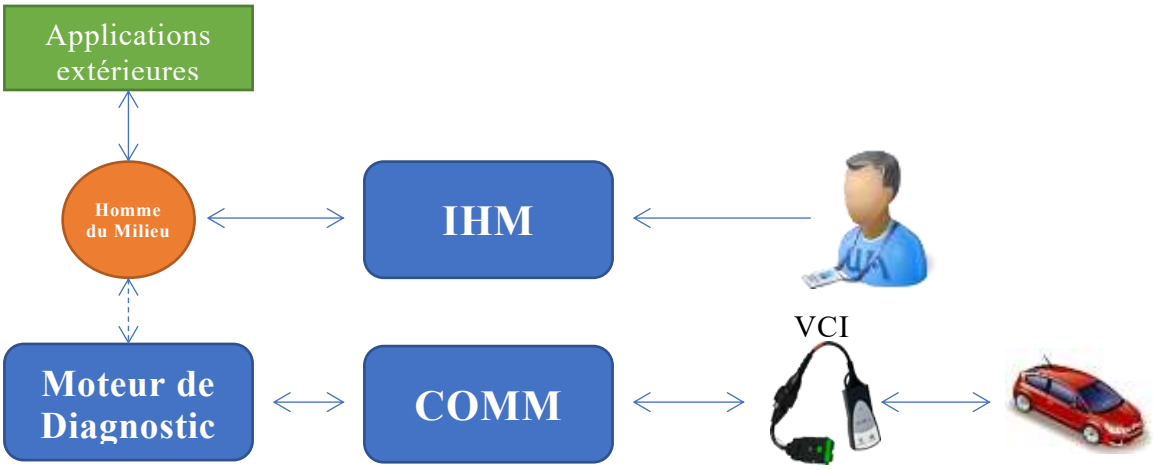


FIGURE 33. INTEGRATION DE L'APPLICATION HOMME DU MILIEU DANS L'ARCHITECTURE DE DIAGBOX AFIN D'ACCEDER ET DE FILTRER CERTAINES INFORMATIONS DEPUIS DES APPLICATIONS EXTERIEURES

5.2.1.3 L'outil de simulation des trames véhicules : Spy And Sim

Comme nous l'avons abordé dans la partie précédente, le logiciel de diagnostic DiagBox a besoin de communiquer avec le véhicule en cours de diagnostic afin de présenter des informations cohérentes tout au long de la navigation. Cette communication s'effectue grâce à des échanges de trames (c'est-à-dire des séries de bit dont le format est dépendant du calculateur et de la propriété interrogée) via un composant physique, la VCI[33], qui est connecté au PC sur lequel est exécuté DiagBox d'un côté (via USB ou Bluetooth), et à la prise OBD[34] du véhicule de l'autre. Ces échanges prennent la forme d'une suite de question/réponses : l'outil de diagnostic va dans un premier temps poser une question au véhicule en fonction de l'opération souhaitée ; le véhicule va ensuite effectuer des actions en interne pour renvoyer une réponse cohérente à l'outil avec l'état actuel du véhicule.

Pour plusieurs raisons pratiques (dont la principale est la validation des fonctionnalités implémentées dans DiagBox), ACTIA a par le passé développé l'outil Spy And Sim (voir Figure 34) qui propose deux modes de fonctionnement différents :

- Dans un premier temps, il est capable d'espionner et d'enregistrer sous un format particulier (que nous appellerons dans cette partie *script*) les échanges de trames et les couples question/réponse associés qui ont lieu entre le véhicule et l'outil DiagBox lors d'une session de diagnostic "réelle" (e.g. en étant connecté physiquement à un véhicule).

[32] https://www.w3.org/TR/wsdl20-primer/

[33] VCI : Vehicle Communication Interface (Interface de Communication Véhicule)

[34] OBD : On Board Diagnosis (Diagnostic embarqué)

- Dans un second temps, il permet d'effectuer des sessions de diagnostic "simulées" (e.g. en n'étant pas connecté physiquement à un véhicule). Pour cela, Spy And Sim s'appuie sur les scripts (et couples question/réponse associés) préalablement construits (soit via la fonctionnalité d'espionnage, soit manuellement) pour aller chercher une réponse possible à une question particulière posée par DiagBox. Notons que la plupart des questions possèdent plusieurs réponses possibles (exemple simplifié : Question = "Est-ce que les feux de croisement sont allumés ?" ; Réponses possibles = "OUI" / "NON"). Dans ce cas, le script choisi arbitrairement l'un des couples question/réponse correspondant et retourne la réponse associée. Afin de contrôler précisément la réponse donnée, il est possible d'activer ou de désactiver chaque couple un par un via l'interface utilisateur de l'outil ; seul les couples activés seront alors examinés pour chercher une réponse adéquate.

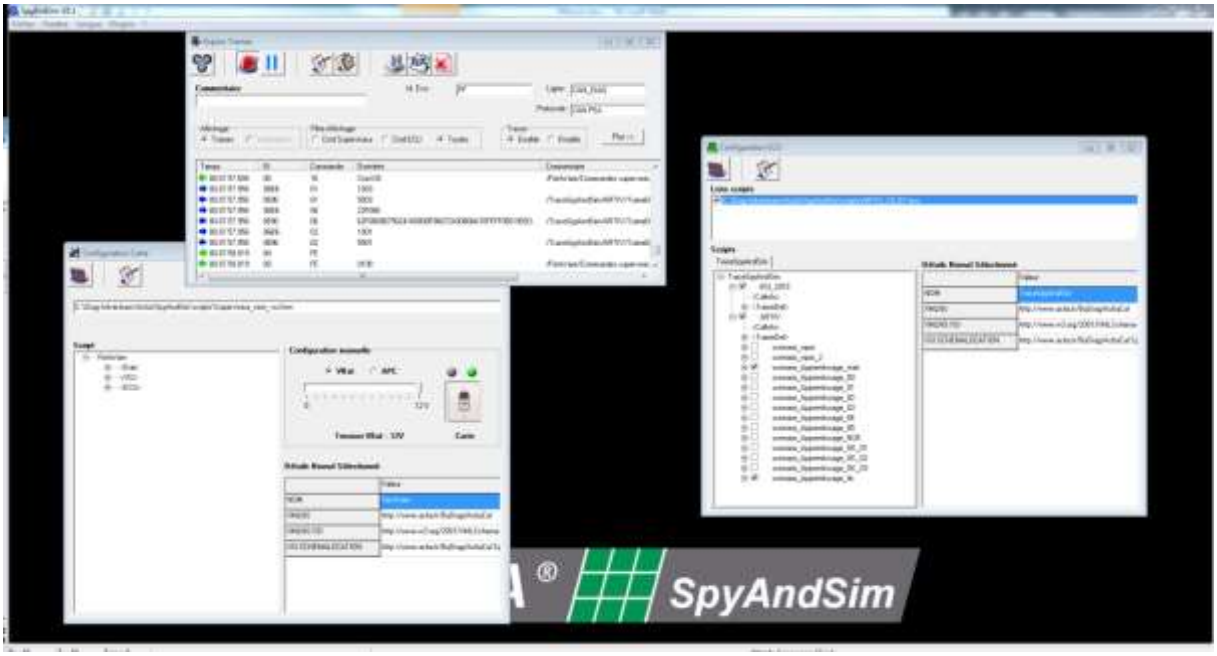


FIGURE 34. CAPTURE D'ECRAN DE SPY AND SIM. L'APPLICATION EST COMPOSEE DE TROIS SOUS-FENETRES. LA PREMIERE (EN HAUT) MONTRE LES ECHANGES DE TRAMES EFFECTUEES ENTRE SPY AND SIM ET DIAGBOX. LA DEUXIEME (EN BAS A GAUCHE) PRESENTE DIFFERENTES INFORMATIONS RELATIVES A LA SIMULATION DE LA VCI. LA TROISIEME (A DROITE) MONTRE LE OU LES SCRIPTS SPY AND SIM QUI SONT CHARGES ET DANS LESQUELS L'APPLICATION VA CHERCHER LES TRAMES A RENVOYER A DIAGBOX DANS LE MODE "SIMULATION"

Dans le cadre des scénarios de formation implémentés tout au long de nos travaux, nous avons donc utilisé Spy And Sim afin de créer les scripts requis pour chaque scénario, puis pour simuler les échanges de trames lors de l'exécution de ces scénarios. Afin de reproduire exactement certains comportements du véhicule pendant la navigation dans DiagBox, nous avons été amenés à décrire l'activation ou la désactivation de plusieurs couples question/réponse en fonction de la progression de l'apprenant dans la leçon. En se référant à la partie 3.2.2.3, nous avons pour cela associé à chaque couple question/réponse un Comportement (l'activation/désactivation d'un Comportement via le scénario entraînant l'activation/désactivation du couple question/réponse associé). Il aurait été possible de décrire des Comportements de plus "haut-niveau" en associant à chacun d'eux plusieurs couples question/réponse ; nous n'avons cependant pas choisi cette solution car nous souhaitions faciliter l'implémentation de notre prototype d'outil de formation en nous servant directement des couples question/réponse mis à disposition dans Spy And Sim.

Finalement, le modèle permettant de représenter l'état de l'outil Spy And Sim et donc du module des Systèmes de Simulation de notre Environnement de Formation est défini par la valeur d'activation (oui ou non) de chacun des couples question/réponse du script associé. A l'exécution des scénarios, l'activation/désactivation de chacun de ces couples est rendue possible grâce à la mise à disposition d'une interface utilisant le format d'échange SOAP de la même manière que pour le logiciel de diagnostic DiagBox.

5.2.1.4 Le Contexte Partagé

Finalement, afin d'interfacer l'Environnement Virtuel, le logiciel de diagnostic DiagBox, et l'outil de simulation des trames véhicules Spy And Sim, et d'avoir un élément unique permettant d'interagir uniformément avec chacun de ces trois modules, nous avons implémenté un système de contexte partagé. Contrairement à ce qui a été présenté dans la partie 3.2.2.4 où le Contexte Partagé fait partie intégrante de l'Environnement de Formation, le système que nous avons développé est directement intégré dans les applications devant échanger des informations avec les modules de l'EF (dans notre cas : le moteur d'exécution des scénarios et l'environnement auteur que nous présenterons respectivement dans les parties 5.2.2 et 5.3). La raison pour cela est que certains développements ont dû être effectués avant que nous ayons terminé de définir précisément l'outil de formation et la méthodologie associée présentés dans la 0. La Figure 35 présente l'intégration effective de notre contexte partagé dans les différents systèmes implémentés.

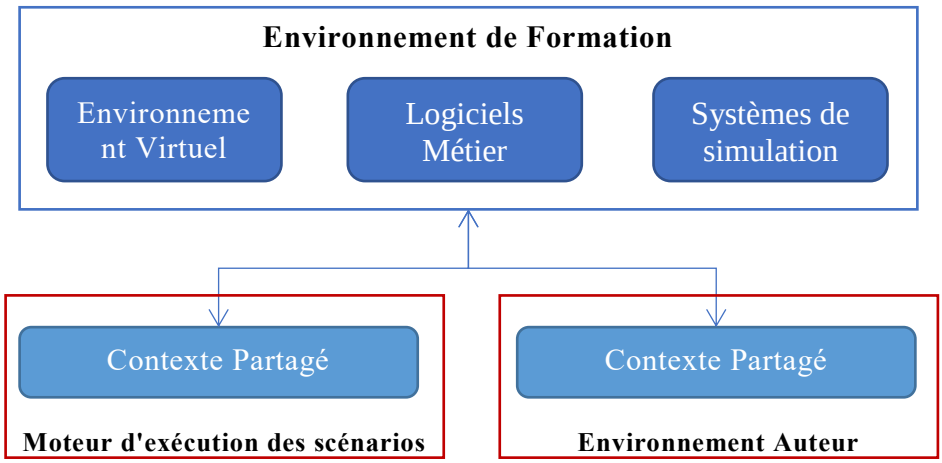


FIGURE 35. INTEGRATION EFFECTIVE DU CONTEXTE PARTAGE DANS NOS APPLICATIONS VIS-A-VIS DE L'ENVIRONNEMENT DE FORMATION

Le Tableau 1 résume les différentes propriétés de chaque module accessible au travers de notre contexte partagé. La première colonne présente la propriété étudiée sur la ligne associée en se basant sur les modèles présentés dans la partie 3.2.2 et appliqués à notre contexte industriel dans les parties 5.2.1.1, 5.2.1.2, et 5.2.1.3. La deuxième colonne identifie la propriété à l'aide d'une chaine de caractères qui sera celle utilisée dans la description des scénarios pour orchestrer l'Environnement de Formation. La troisième et dernière colonne indique si la valeur de la propriété est modifiable manuellement depuis des applications extérieures (une requête de modification pourra toujours être envoyée au contexte partagé, mais elle sera ignorée si la modification n'est pas possible). Précisons enfin que la valeur de chaque propriété est toujours

observable, et que tout changement de valeur est notifié afin que les applications extérieures puissent réagir en conséquence. De plus, certaines propriétés qui n'ont pas de correspondance directe dans le module associé ont été rajoutées dans le contexte partagé par rapport à nos besoins initiaux. Nous avons effectué cela afin de correspondre au fonctionnement de notre environnement d'exécution des scénarios (voir partie 5.2.2 pour plus d'explications).

TABLEAU 1. PROPRIETES DE L'ENVIRONNEMENT DE FORMATION ACCESSIBLES DEPUIS LE CONTEXTE PARTAGE

Propriété de l'EF	Identificateur	Altérable
EV : Environnement courant	ve.environment	Oui
EV : Point de vue courant dans de l'Environnement X	ve.x.pov	Oui
EV : accessibilité du Point de vue Y dans l'environnement X	ve.x.pov.y.is_available	Oui
EV : dernière Interaction effectuée	ve.interaction	Non
EV : Interactions autorisées	ve.allowed_interactions	Oui
EV : Interaction à simuler	ve.interaction_to_simulate	Oui
EV : visibilité de l'Elément Y dans l'environnement X	ve.x.element.y.is_visible	Oui
EV : interactivité de l'Elément Y dans l'Environnement X	ve.x.element.y.is_interactive	Oui
EV : Valeur de l'Etat Z de l'Elément Y dans l'Environnement X	ve.x.element.y.state.z	Oui
EV : disponibilité de l'Interaction Z de l'Elément Y dans l'Environnement X	ve.x.element.y.interaction.z	Oui
DiagBox : Ecran courant de l'onglet X	dbx.x.screen	Non
DiagBox : Popup courante de l'onglet X	dbx.x.popup	Non
DiagBox : dernière Action Utilisateur effectuée	dbx.user_action	Non
DiagBox : Actions Utilisateurs autorisées	dbx.allowed_user_actions	Oui

DiagBox : Action Utilisateur à simuler	dbx.user_action_to_simulate	Oui
Spy And Sim : Activation du comportement X	sns.x	Oui

Finalement, les applications extérieures peuvent interagir de manière uniforme avec chaque propriété du contexte partagé de trois manières :

- en étant notifiées d'un changement de valeur de la propriété.
- en lisant la valeur de la propriété.
- en modifiant la valeur de la propriété.

C'est ensuite le rôle du contexte partagé que de diriger chaque requête (ou réponse de requête dans le cas des notifications) vers le module correspondant. De plus, l'implémentation de contexte partagé a été réalisée dans différents langages du fait de la conception adoptée (voir Figure 35) : C++ pour l'environnement d'exécution des scénarios, et JAVA pour l'environnement auteur. Enfin, les échanges entre le contexte partagé et les différents modules de l'EF sont dans chaque cas effectués en utilisant la technologie appropriée (protocole TCP/IP pour l'Environnement Virtuel, format SOAP pour DiagBox et Spy And Sim).

5.2.2 Moteur d'exécution des Scénarios de Formation

Afin de mettre en œuvre l'ensemble des points présentés dans le Chapitre 3, nous avons implémenté un environnement informatique dont le rôle est :

- d'exécuter les scénarios pour instrumenter l'Environnement de Formation et apporter les éléments pédagogiques et ludiques en fonction des progrès de l'apprenant (5.2.2.1) ;
- de présenter et d'organiser l'Environnement de Formation et ses différents modules (5.2.2.2) ;
- de présenter les différents éléments pédagogiques et ludiques (5.2.2.3).

Cet environnement, que nous appelons moteur d'exécution des scénarios, vient donc se greffer autour des deux grands éléments qui composent notre outil de formation : l'Environnement de Formation et le Scénario. Nous l'avons implémenté en utilisant le langage C++ ainsi que plusieurs librairies que nous présenterons tout au long de cette partie.

5.2.2.1 Exécuter les scénarios

Afin de fonctionner, notre moteur d'exécution prend en paramètre d'entrée le chemin désignant le répertoire dans lequel se trouvent toutes les données relatives au scénario souhaité. Ce répertoire doit contenir :

- Un fichier *[nom_du_scénario].xml*, qui contient des informations générales liées au scénario.
- Un fichier *[nom_du_scénario].as*, qui contient la description du scénario dans un format différent du langage dédié que nous avons défini dans le 0. En effet, notre langage dédié permet de décrire les différents concepts nécessaires pour écrire une leçon se déroulant dans notre Environnement de Formation, mais chacun est libre de les interpréter comme

il le souhaite afin d'obtenir le résultat adapté lors de l'exécution du scénario. Le format utilisé est celui défini par la librairie AngelScript[35] (proche du langage C/C++), qui est une plateforme permettant d'étendre et de personnaliser les fonctionnalités d'applications écrites en C++ à l'aide de scripts. Afin d'obtenir ce fichier, nous avons implémenté à l'aide de l'environnement ANTLRWorks 2 [36] une application JAVA permettant d'interpréter notre langage dédié et de le transformer vers le format AngelScript souhaité.

- Un répertoire *database*, qui contient :
  - un ensemble de fichiers décrivant les objectifs pédagogiques et les retours pédagogiques associés au scénario. Notons par ailleurs l'absence de fichiers décrivant les différents éléments ludiques du scénario du fait que notre implémentation actuelle ne les gère pas (voir partie 5.2.2.3) ;
  - un fichier décrivant l'état initial de l'Environnement Virtuel ainsi que les objets factices ajoutés (voir partie 5.3.3).

Après avoir exécuté les différents modules de l'Environnement de Formation, notre moteur d'exécution vient charger l'ensemble des éléments pédagogiques et initialiser l'Environnement Virtuel. Puis, nous chargeons le script décrivant notre scénario et l'exécutons dans un Thread dédié à l'aide de la librairie AngelScript.

De la même manière que dans le Chapitre 3 et le Chapitre 4, nous nous sommes concentrés sur la gestion des scénarios de type "guidé". De plus, des contraintes liées à l'utilisation de DiagBox (comportement non-défini dans le cas où Spy And Sim ne trouve pas de réponse à une question particulière) ainsi qu'aux besoins de notre client nous ont dirigés vers une interprétation des scénarios où l'apprenant ne peut pas effectuer une autre action que celle(s) attendue(s). Pour cela :

- notre contexte partagé possède deux propriétés supplémentaires liées respectivement à DiagBox et à l'Environnement Virtuel : *dbx.allowed_user_actions* et *ve.allowed_interaction* (voir 5.2.1.4). La valeur associée à ces propriétés est une liste des interactions autorisées dans le module correspondant à un instant précis du scénario. Ainsi, lorsque notre contexte partagé est notifié d'une interaction, celui-ci vérifie automatiquement que cette interaction est autorisée. Si oui, une réponse positive est retournée via l'interface d'échange entre le contexte partagé et le module associé. Si non, une réponse négative est retournée, et les développements effectués pour DiagBox et l'EV nous permettent de bloquer cette interaction (voir 5.2.1.1 et 5.2.1.2).
- l'application qui interprète et convertit notre langage dédié vers le langage AngelScript traite les instructions `Checkpoint` portant sur les propriétés *dbx.user_action* et *ve.interaction* en deux temps. Tout d'abord, une instruction intermédiaire est créée afin de modifier les propriétés *dbx.allowed_user_actions* et *ve.allowed_interaction* de manière adéquate. Puis, une boucle est créée afin de vérifier cycliquement les conditions liées à l'instruction `Checkpoint` traitée.

Finalement, notre moteur d'exécution implémente un comportement fortement contraint dans la gestion de nos scénarios, et donc assez peu ludique. Comme évoqué précédemment, cela a été réalisé de cette manière afin de répondre à des contraintes imposées à la fois par notre client et

[35] http://www.angelcode.com/angelscript/

[36] http://tunnelvisionlabs.com/products/demo/antlrworks

par le logiciel de diagnostic. Nous prévoyons cependant de continuer la réflexion afin d'être en mesure de rendre notre outil d'exécution plus libre et donc plus ludique (voir Perspectives).

5.2.2.2 Présenter et organiser l'Environnement de Formation

Afin que l'apprenant puisse manipuler de manière ergonomique les différents modules de l'Environnement de Formation, le moteur d'exécution des scénarios que nous avons développé utilise une mise en page spécifique illustrée sur la Figure 36 et la Figure 37.

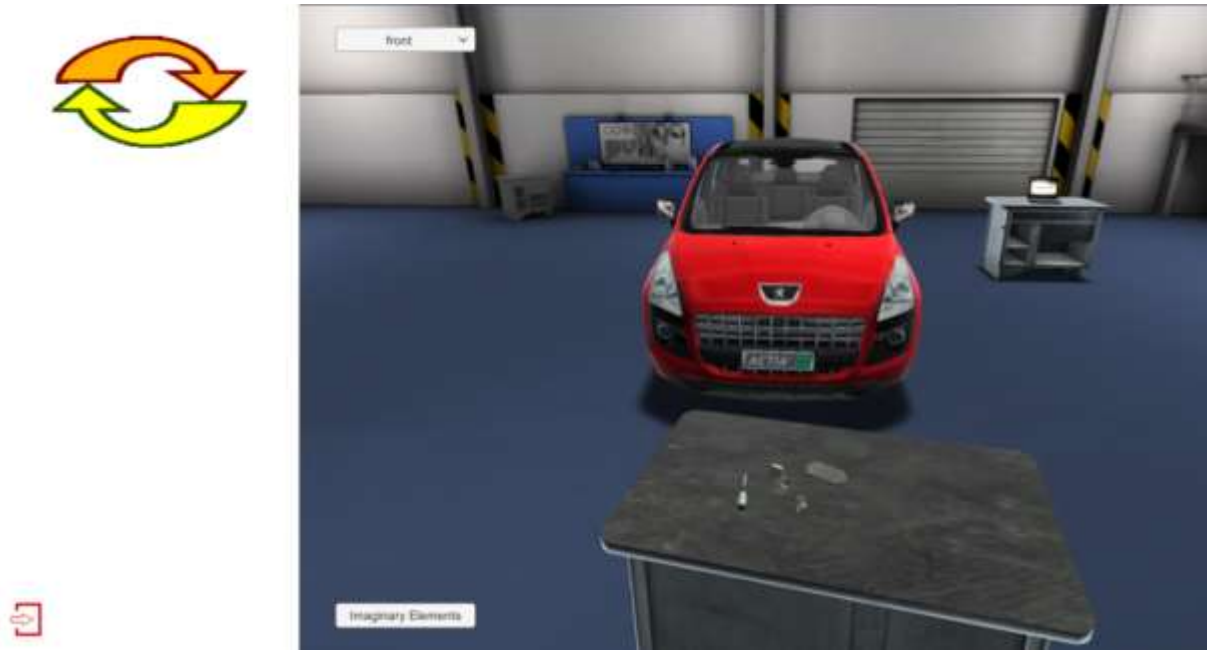


FIGURE 36. MOTEUR D'EXECUTION DES SCENARIOS : VUE ENVIRONNEMENT VIRTUEL

Nous observons les différents éléments qui composent notre mise en page :

- Le ***bandeau***. Sur la gauche de l'écran est affiché un bandeau dont le rôle est de présenter un certain nombre d'éléments relatifs à la leçon. Il introduit aussi la vue secondaire en haut (voir plus loin) et le bouton quitter en bas.
- La vue ***principale***. La partie droite de l'écran est réservée à l'affichage d'un module parmi ceux dans lesquels l'apprenant doit évoluer pour progresser dans la leçon. (DiagBox ou l'Environnement Virtuel). Sur le module actuel de la vue principale, l'utilisateur peut donc interagir normalement.
- La vue ***secondaire***. Le haut du bandeau est réservé à l'affichage d'une capture d'écran du module qui n'est pas couramment affiché dans la vue principale. Il n'est donc pas possible pour l'utilisateur d'interagir sur cette vue ; cependant, il peut cliquer dessus afin d'interchanger le module de la vue principale et le module de la vue secondaire.

Le module de Spy And Sim est quant à lui caché à l'utilisateur, car il est entièrement piloté par le scénario et l'apprenant n'a donc pas besoin d'interagir avec lui pour progresser dans la leçon. Plus généralement, cette mise en page a été choisie pour respecter trois contraintes industrielles :

- la résolution minimale supportée par DiagBox est 1024 x 768 ;
- la résolution maximale des ordinateurs sur lesquels sont installés DiagBox est 1366x768 ;

- l'outil de formation doit pouvoir être exécuté sur ces ordinateurs.

Ainsi, le seul espace possible pour fournir des informations extérieures aux différents modules de l'EF est un bandeau dont la largeur correspond à la différence entre la largeur maximale de l'écran et la largeur minimale de DiagBox. Finalement, les différents modules de l'Environnement de Formation sont exécutés chacun de leur propre fenêtre, mais afin que la mise en page adoptée et le bandeau de gauche en particulier permette d'organiser et d'unifier leur présentation et leur utilisation.

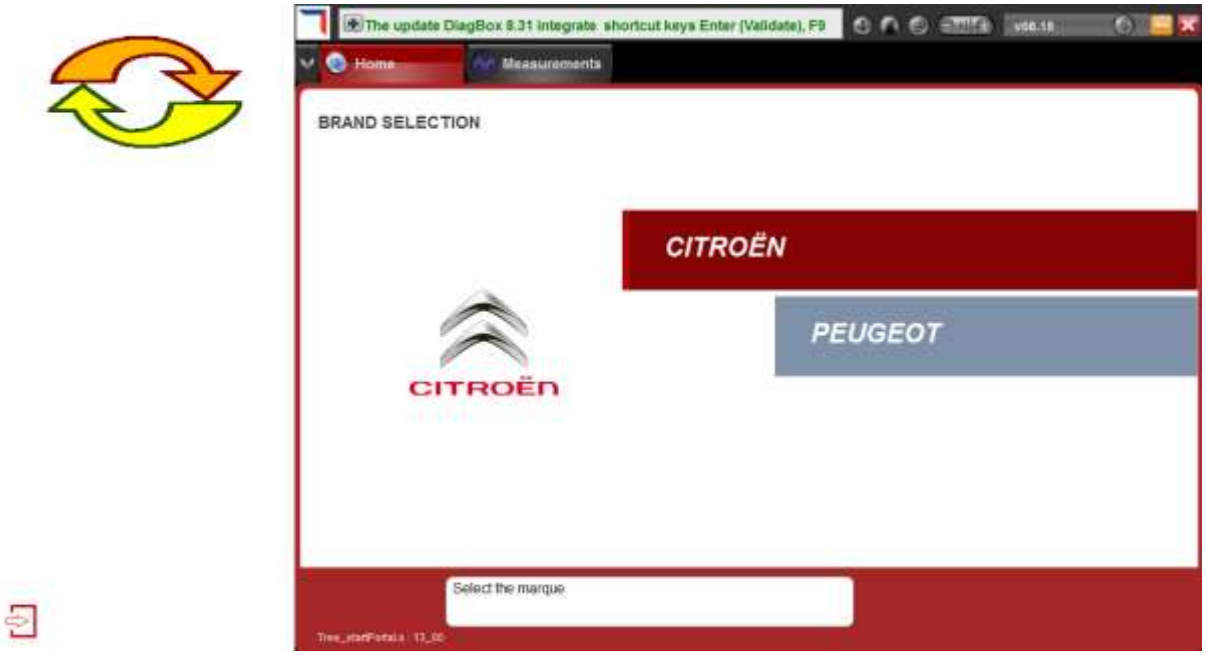


FIGURE 37. MOTEUR D'EXECUTION DES SCENARIOS : VUE DIAGBOX

5.2.2.3 Présenter les éléments pédagogiques et ludiques

Dans le cadre de son implémentation actuelle, notre moteur d'exécution des scénarios se limite à l'utilisation de deux types de retours pédagogiques qui sont présentés à l'apprenant via le bandeau introduit précédemment (voir Figure 38) :

- les messages d'accompagnement, dont le rôle est d'indiquer de manière générale l'étape qu'est en train d'accomplir l'apprenant. Ces messages sont comparables à l'intitulé des différentes sous-missions de la leçon (voir partie 3.2.3.2), mais ils sont décrits dans le scénario comme étant des retours pédagogiques (utilisation de l'instruction `PedagogicalFeedback`). Dans le bandeau, le message de support courant est affiché de manière statique juste en dessous de la vue secondaire.
- les messages d'aide, qui permettent d'indiquer exactement ce que doit réaliser l'apprenant afin de progresser dans la leçon. Dans le bandeau, l'aide courante est accessible en cliquant sur le bouton associé. Le message correspondant apparait alors statiquement dans un encadré bleu. De plus, le bouton d'aide se met à clignoter si l'utilisateur effectue trop d'actions (trois actuellement) qui ne sont pas correctes afin de progresser dans la leçon, ou si trop de temps (quinze secondes actuellement) s'écoule sans que l'utilisateur n'ait effectué l'action attendue. Cette fonctionnalité a été implémentée afin d'attirer l'attention de l'utilisateur lorsqu'il semble bloqué à un moment particulier du scénario,

mais n'est aucunement obligatoire afin de lui laisser l'opportunité de continuer à chercher la solution par lui-même.

- les messages d'erreur, qui permettent de donner des informations supplémentaires à l'apprenant lorsqu'il effectue une action non attendue. Dans le bandeau, ils apparaissent sur un fond rouge à l'instant où l'apprenant commet l'erreur, en lieu et place du message d'accompagnement. Le message d'erreur disparaît alors progressivement afin de réafficher ce message d'accompagnement.

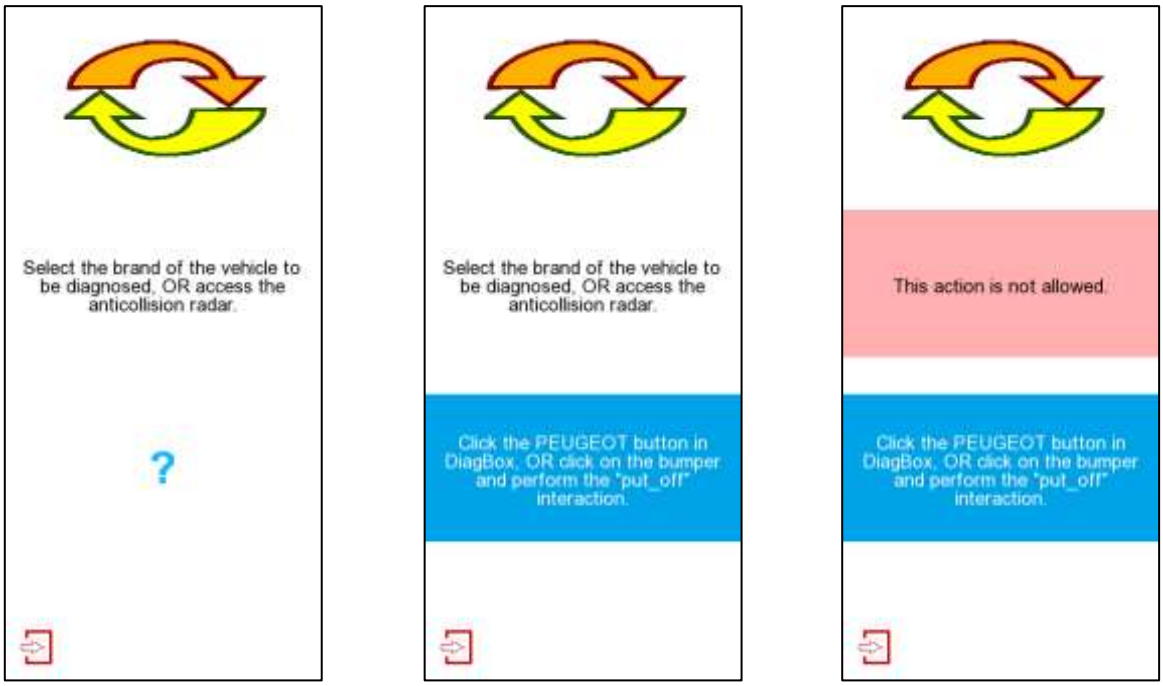


FIGURE 38. PRESENTATION DES DIFFERENTS ELEMENTS PEDAGOGIQUES GERES PAR NOTRE MOTEUR D'EXECUTION

Dans l'objectif de développer un outil fonctionnel rapidement, nous nous sommes finalement limités à l'implémentation de ces trois types d'éléments pédagogiques. Le compte-rendu de la leçon est lui aussi absent de notre implémentation. De plus, aucun élément ludique n'a été intégré. Bien sûr, nous prévoyons d'améliorer ces points-là dans le futur afin d'augmenter les possibilités offertes par notre outil (voir les Perspectives).

5.2.3 Évaluation

Nous présentons dans cette partie la phase d'évaluation que nous avons effectuée afin de valider la pertinence de l'outil de formation proposé puis implémenté. Nous présentons alors en 5.2.3.1 le projet Diag'Adventures qui a été mené en parallèle des travaux présentés dans ce mémoire. L'objectif principal de ce projet était de parvenir à un prototype d'outil de formation dont les caractéristiques sont similaires à celles introduites dans le 0, afin éventuellement d'en faire la démonstration à des clients par la suite. Du fait que le prototype ainsi développé soit industriellement et visuellement parlant plus abouti que l'outil de formation que nous avons implémenté dans le cadre de cette thèse, c'est lui qui a été mis à disposition des garagistes pour ensuite recueillir leurs impressions (5.2.3.2).

5.2.3.1 Le projet Diag'Adventures

En parallèle des travaux réalisés dans cette thèse, plusieurs développements ont été menés dans le cadre du projet Diag'Adventures réalisé dans le cadre d'un partenariat entre :

- le Serious Game Research Lab (SGRL) de l'université Jean-François Champollion à Albi[37]. Il s'agit d'un laboratoire dont l'équipe travaille sur plusieurs projets de Jeux Sérieux.
- la société Opérantis, basé à Saint-Jean à côté de Toulouse[38]. C'est une société spécialisée dans le développement d'outils de formation innovants, tels que les Jeux Sérieux ou des environnements de simulation complexes.
- la division Automotive du groupe ACTIA, dont le siège social se situe à Toulouse[39]. L'objectif global de l'entreprise est la conception et la production de systèmes électroniques liés aux domaines de l'Automotive et des Télécommunication. En particulier, la division Automotive se concentre sur les systèmes embarqués et les outils de diagnostic véhicule.

Ce projet, pleinement soutenu par la région Midi-Pyrénées, avait dans un premier temps pour objectif le développement d'un prototype d'outil de formation de type Jeu Sérieux pour l'enseignement de procédures métiers complexes dans le contexte industriel du diagnostic automobile. Dans ce cadre, plusieurs scénarios de formation ont été implémentés, tel que le diagnostic d'une panne de feux de croisement, l'apprentissage de la nouvelle fonctionnalité de télédiagnostic, ou encore le réglage du radar ARTIV (dont le début a été présenté dans la partie 3.3.2). Dans un second temps, l'objectif recherché était l'évaluation de la pertinence liée à l'utilisation de ce genre d'outil afin d'accompagner les garagistes dans leurs formations en présentiel et à distance.

Le projet Diag'Adventures a donné naissance à plusieurs maquettes d'outil de formation avant de parvenir à une version satisfaisante qui puisse être déployée dans un ensemble de garages et ensuite être évaluée par les garagistes eux-mêmes. Finalement, cet outil de formation présente les mêmes fonctionnalités que celui introduit dans la partie 5.2.2, avec cependant quelques différences et améliorations (avec principalement l'ajout d'éléments ludiques), illustrées en Annexe 5 :

- L'Environnement Virtuel est implémenté en 3D ; la principale différence réside dans le fait que l'utilisateur déplacer la caméra librement dans la simulation de son environnement de travail habituel.
- Le carnet de bord. Au début du scénario, un carnet de bord est présenté à l'apprenant afin d'introduire les grandes étapes (= sous-missions) qu'il va devoir accomplir. Par la suite, l'apprenant peut ouvrir ce carnet de bord à tout instant afin de suivre sa progression dans la leçon en temps réel.
- Le bandeau de gauche affiche maintenant :
    - la sous-mission actuelle en haut du bandeau ;

[37] https://www.univ-jfc.fr/equipesrecherche/serious-game-research-lab

[38] http://www.operantis.fr/societe/qui-sommes-nous/

[39] http://actia.com/fr/groupe/profil/division-automotive

- le message d'accompagnement en dessous de la sous-mission. Ici, ce message est beaucoup plus spécifique que dans notre moteur d'exécution. Il indique en fait ce que doit globalement réaliser l'apprenant, sans préciser l'action unitaire dans l'Environnement Virtuel ou DiagBox qu'il doit effectuer.
- un chronomètre afin que l'apprenant puisse connaitre le temps passé à accomplir le scénario. Notons que ce chronomètre n'est présent qu'à but informatif et n'a pas pour objectif d'évaluer l'apprenant. En effet, il ne faut pas que le scénario de formation prenne trop de temps à être effectué sur le temps de travail des garagistes. Nous nous sommes donc servis de cette information pour plus tard apporter quelques améliorations à l'outil et au scénario en particulier.
- à côté du bouton d'aide, un indicateur présentant le nombre de fois que l'aide a été demandée par l'apprenant. De la même manière que le chronomètre, l'objectif de cet indicateur est strictement informatif.

- Les messages d'aide, toujours accessibles via le bouton correspondant dans le bandeau, sont ici présentés dans une pop-up.
- Il est possible d'apporter des messages d'information tout au long du scénario afin, par exemple, d'apporter des informations supplémentaires n'ayant pas pu être représentées dans le scénario. De la même manière que les messages d'aide, ces informations sont apportées dans une pop-up.
- Il est possible de poser des questions à l'utilisateur afin de vérifier ses connaissances sous formes de quiz. Ces quiz peuvent comporter du texte et/ou des images.
- Il est possible de mettre en scène des dialogues dans l'Environnement Virtuel entre un garagiste et un client, afin de contextualiser le scénario.

Sur la base de ce prototype, plusieurs scénarios de formation ont été implémentés afin d'illustrer le concept à notre client. Finalement, le scénario ayant été choisi pour être évalué par les garagistes est le scénario de réglage du radar anticollision introduit en 3.3.2 (et présenté dans sa totalité en Annexe 1). Ce scénario a été sélectionné du fait de la complexité qu'il présente. En effet, en dehors des manipulations requises sur le véhicule afin de régler manuellement l'orientation du radar, la procédure associée requiert une phase d'apprentissage sur route afin que DiagBox puisse calculer le degré de désorientation du radar, permettant ensuite de fournir au garagiste les données nécessaires au réglage. Cette phase d'apprentissage demandant de respecter des conditions très particulières, il est très difficile et coûteux de la mettre en place lors des sessions de formation classique (cours + travaux pratiques) données aux garagistes chaque année. L'utilisation de notre outil de formation pour l'apprentissage de cette procédure est donc apparue pertinente aux yeux de notre client.

5.2.3.2 Tests terrain et Résultats

Afin d'évaluer la pertinence de l'outil de formation proposé dans ce manuscrit, c'est-à-dire la forme qu'il prend et son utilité réelle, nous avons procédé à deux phases successives de tests terrain. Pour chaque phase, nous avons sélectionné un ensemble de garages-cibles sur la région parisienne et toulousaine. Nous nous sommes alors rendus dans chacun de ces garages pour installer le prototype sur les machines utilisées par les garagistes, puis :

- lors de la première phase, nous leur avons fourni de visu des explications sur le fonctionnement de l'outil. Puis nous leur avons laissé une durée de 3 mois pour effectuer autant de fois qu'ils le souhaitaient le scénario de formation et remplir le questionnaire associé.
- lors de la deuxième phase, nous leur avons fourni de visu des explications sur le fonctionnement de l'outil, puis leur avons demandé d'effectuer le scénario et de remplir le questionnaire dans le prolongement de ces instructions initiales. En effet, lors de la première phase de test, nous avons eu des difficultés à obtenir des réponses au questionnaire de la part des garagistes du fait de leur emploi du temps chargé.

En dehors des questions classiques permettant d'établir le profil des personnes ayant testé notre outil, le questionnaire que nous leur avons demandé de remplir (disponible en Annexe 6) avait pour objectif d'évaluer cinq grands points :

- l'appréciation globale de l'outil de formation.
- l'immersion dans l'Environnement de Formation et sa pertinence.
- la pertinence des apprentissages apportés au travers de notre outil, en comparaison avec une session de formation plus "classique".
- la difficulté du scénario proposé.
- l'intérêt général pour ce type d'outil de formation.

Lors de la première phase de test, nous avons fait intervenir 6 garages du réseau PSA (4 sur la région parisienne et 2 sur la région toulousaine). Cela nous a permis de recueillir les impressions de 13 garagistes (aux profils variés) par rapport à chacun des critères présentés ci-dessus (voir deuxième colonne du Tableau 2).

L'ergonomie de l'outil de formation et plus particulièrement de l'Environnement Virtuel. En effet, dans le cas du prototype implémenté pour le projet Diag'Adventures, l'EV propose plusieurs types d'interaction, dont certaines se basent sur la manipulation plus ou moins complexe d'objets dans l'environnement 3D (ex : rotation du tournevis). Certaines personnes ont donc eu des difficultés à progresser dans le scénario du fait que ces interactions ne soient pas adaptées à des profils n'ayant pas l'habitude d'évoluer dans ce type d'environnement.

- La précision des aides apportées tout au long du scénario. En effet, du fait que le scénario ait été implémenté par des personnes ne possédant pas l'expertise des formateurs en termes de pédagogie, les retours pédagogiques définis n'étaient pas forcément très précis ni pertinents.

Avant d'effectuer la deuxième phase de test, nous nous sommes concentrés sur l'amélioration de ces deux points en ne proposant qu'un seul type d'interaction sous forme de menu déroulant, et en apportant des retours pédagogiques plus adaptés. De plus, nous avons implémenté un scénario "tutoriel" permettant d'introduire les différents types d'interaction que l'apprenant sera amené à effectuer dans le scénario évalué. Nous avons alors fait intervenir 4 nouveaux garages afin de recueillir les impressions sur la version de notre outil qui prend en compte les évolutions effectuées. Cette fois, uniquement 6 garagistes ont participé à la phase de test, nous permettant alors d'obtenir les évaluations présentées dans la troisième colonne du Tableau 2.

Finalement, nous constatons une nette augmentation dans les notes attribuées aux différents critères que nous souhaitions évaluer. Notre outil de formation a été bien mieux reçu que lors de la première phase d'évaluation, confirmant ainsi les axes d'améliorations que nous avions définis.

Ainsi, 84% des personnes interrogées se disent intéressées par l'introduction de ce genre d'outil au sein des formations existantes ou des futures formations. En outre, l'intégration de l'outil de diagnostic directement dans l'Environnement de Formation leur a semblé pertinente car il représente l'outil qu'ils utilisent quotidiennement et ils n'ont alors pas à se poser de questions supplémentaires lorsqu'ils passent du scénario de formation au cas réel. Cela nous a alors permis de valider la forme de l'outil de formation proposé au travers du projet Diag'Adventures, et plus généralement dans le cadre de cette thèse.

TABLEAU 2. RESULTATS DES DEUX PHASES DE TESTS EFFECTUEES POUR EVALUER LA PERTINENCE DE NOTRE OUTIL DE FORMATION

	Test n°1	Test n°2	Différence
Nombre de réponses	14 (3 mois)	6 (2 semaines)	-
Appréciation globale /10	5.9	8.6	+ 2.7
Immersion /10	6.1	9.3	+ 3.2
Pertinence /10	5.8	8.8	+ 3.0
Difficulté /10	5.7	9.0	+ 3.2
Intéressés ?	71% (10 Oui, 3 Non, 1 Neutre)	84% (5 Oui, 1 Neutre)	+ 13 points

5.3 Environnement Auteur

Le deuxième grand système qui a été implémenté lors de nos travaux concerne un environnement permettant aux formateurs de créer leurs propres scénarios de formation en suivant la méthodologie décrite dans le Chapitre 4. Nous décrivons dans cette partie les points essentiels de l'environnement auteur ainsi développé. Après en avoir effectué une présentation générale en 5.3.1, nous décrirons l'implémentation de la base de données pédagogique (5.3.2) pour ensuite nous intéresser à la gestion du modèle de l'Environnement Virtuel (5.3.3). Nous terminerons par un compte-rendu de l'évaluation de notre environnement auteur en 5.3.4.

5.3.1 Présentation générale

L'environnement auteur que nous avons développé (l'environnement TAASER[40]) se présente sous la forme d'une application autonome développée en Java avec les librairies Swing pour la création des interfaces utilisateurs et mxGraph[41] pour la gestion des différentes représentations graphiques proposées par notre méthodologie. Au lancement de l'application (Figure 39), l'utilisateur a le choix entre :

[40] TAASER = a Tool for Automated Authoring of SERious games scenarios

[41] https://jgraph.github.io/mxgraph/

- éditer un ***nouveau scénario***. Dans ce cas, une fenêtre de dialogue apparait (Figure 40) lui demandant d'indiquer le nom du scénario à créer et l'emplacement où il sera sauvegardé.
- éditer un ***scénario existant***. Dans ce cas, une fenêtre de dialogue apparait (Figure 41) lui demandant d'indiquer l'emplacement du fichier principal du scénario (fichier *.taaser*, nous y reviendrons plus tard lorsque nous décrirons la sauvegarde des scénarios).

La validation de l'une des deux fenêtres de dialogue entraine alors l'affichage de l'interface principale de notre environnement (Figure 42 et Figure 43).

FIGURE 39. CAPTURE D'ECRAN DE L'ENVIRONNEMENT AUTEUR AU LANCEMENT

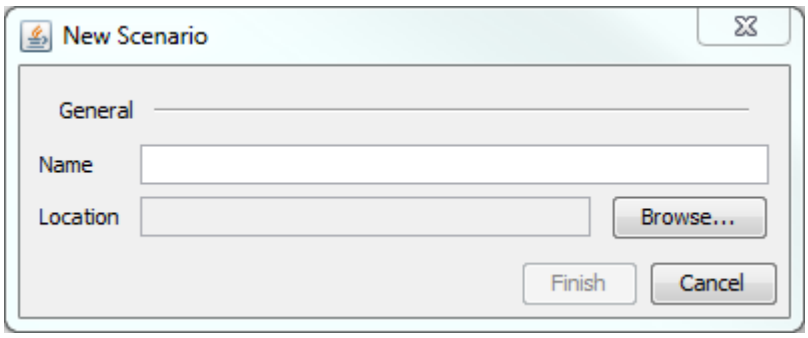


FIGURE 40. FENETRE DE DIALOGUE POUR LA CREATION D'UN NOUVEAU SCENARIO

Cette interface présente plusieurs éléments et fonctionnalités associées :

- La barre des menus. Elle permet à l'utilisateur d'effectuer plusieurs actions réparties dans plusieurs menus :
    - Le menu *Fichier* permet de créer un nouveau scénario ou de charger un scénario existant, de fermer le scénario courant, de sauvegarder le scénario courant et de l'exporter, ou encore de quitter l'environnement auteur. Toutes ces actions sont décrites dans la partie actuelle.

- Le menu *Base de données* permet d'accéder :
  - à l'ensemble des éléments pédagogiques qui sont ou seront utilisés dans le scénario courant (voir partie 5.3.2).
  - au modèle de l'Environnement Virtuel tel qu'il sera utilisé à l'exécution du scénario (voir partie 5.3.3).
- Le menu *Outils* permet :
  - d'accéder aux différents outils de capture permettant la génération automatisée de la description de l'Activité. L'utilisateur peut ainsi exécuter l'Environnement de Formation et ses différents modules (en particulier l'Environnement Virtuel et DiagBox), puis activer la fonctionnalité de "Capture" pour ensuite enregistrer des séquences d'actions effectuées directement dans ces modules.
  - de mettre en forme automatiquement la description de l'Activité, c'est-à-dire d'organiser la représentation graphique correspondante afin qu'elle soit lisible.
- Le menu *Fenêtre* permet simplement de gérer l'affichage des différentes fenêtres de dialogue dans notre application.

- La vue principale. Elle permet à l'utilisateur de décrire dans les onglets correspondants les différentes dimensions du scénario (Activité et Pédagogie) à l'aide des représentations graphiques associées décrites dans la partie 4.2. Notons qu'il n'y a pas d'onglet pour la description des éléments ludiques car cette dimension n'était pas la plus importante aux yeux de notre client. De la même manière que pour notre moteur d'exécution des scénarios, nous avons donc mis l'accent sur l'implémentation des fonctionnalités liées aux deux autres dimensions.
- La fenêtre de dialogue présentant l'ensemble des blocs d'activité disponibles pour décrire le déroulement de la leçon (uniquement présente lorsque l'utilisateur est en train de décrire l'Activité). Ces différents blocs sont classés par catégorie (Environnement Virtuel, DiagBox, …), puis par type (*checkpoint*, *animation*, etc.) afin que l'utilisateur trouve rapidement ceux qui l'intéressent. Le formateur peut ensuite les utiliser dans la description de l'Activité grâce à une opération classique de glisser-déposer vers la vue associée.
- La fenêtre d'inspection des instances de blocs d'activité utilisées dans la description de l'Activité (uniquement présente lorsque l'utilisateur est en train de décrire l'Activité). Elle permet :
  - d'accéder aux différentes propriétés de l'instance de bloc actuellement sélectionnée dans la description de l'Activité et de modifier leur valeur afin de l'adapter au contexte souhaité ;
  - d'accéder aux différents retours pédagogiques associés à l'instance de bloc actuellement sélectionnée dans la description de l'Activité. L'utilisateur peut alors ajouter/supprimer autant de retours pédagogiques qu'il souhaite, et associer à chaque retour pédagogique l'identifiant du retour qui sera effectivement exécuté pendant le déroulement du scénario (en passant par la base de donnée pédagogique).
- La fenêtre d'inspection des blocs d'activité pédagogique et d'objectif pédagogique (uniquement présente lorsque l'utilisateur est en train de décrire la pédagogie. Elle permet :

- de modifier l'intitulé de chaque activité pédagogique et de chaque objectif pédagogique ;
- d'accéder aux différents retours pédagogiques associés au bloc d'activité pédagogique actuellement sélectionné dans la description de la Pédagogie. Les opérations alors possibles sont les mêmes que pour la fenêtre d'inspection des instances de blocs d'activité utilisées dans la description de l'Activité (voir point précédent).

A tout instant de la description du scénario, l'auteur est libre de sauvegarder son scénario afin de reprendre plus tard. Lorsqu'il choisit de faire ainsi en cliquant sur l'action correspondante dans la barre des menus, notre environnement auteur crée (à l'emplacement indiqué lors de la création du scénario) :

- un fichier *.taaser* qui contient plusieurs informations générales liées au scénario ;
- un ensemble de fichiers qui contiennent la description des différentes représentations graphiques du scenario sous un format XML ;
- un répertoire *database* qui contient un ensemble de fichiers décrivant les différents éléments pédagogiques utilisés dans le cadre du scénario sauvegardé.

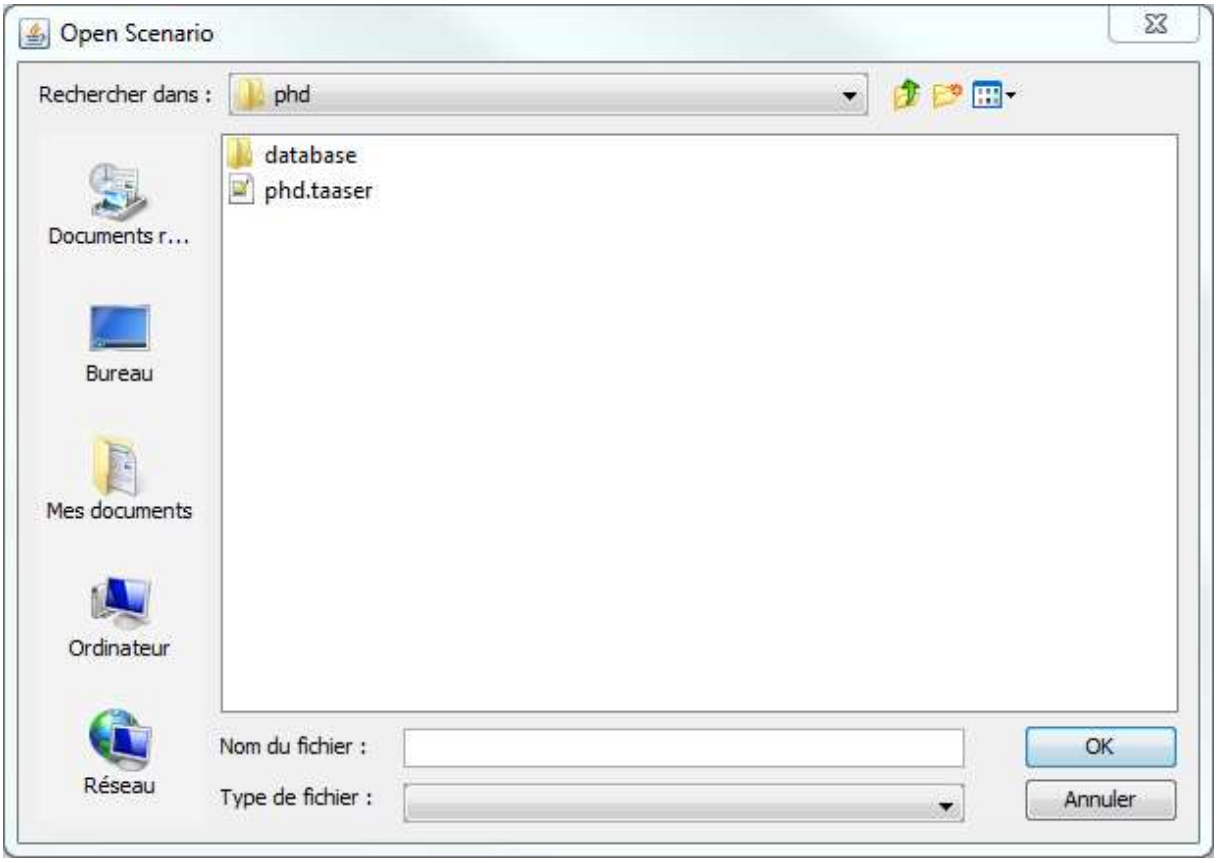


FIGURE 41. FENETRE DE DIALOGUE POUR L'OUVERTURE D'UN SCENARIO EXISTANT

L'auteur peut aussi choisir d'exporter son scénario dans le format défini par notre langage dédié. Dans ce cas, une fenêtre de dialogue apparait (Figure 44) pour indiquer le résultat de la validation du langage dédié (voir partie 4.2.2.1). Si le scénario est effectivement validé, il est d'abord converti vers le langage dédié, puis un répertoire contenant toutes les données nécessaires

à son exécution est créé (voir partie 5.2.2.1). Dans le cas contraire, un message indique la provenance de l'erreur.

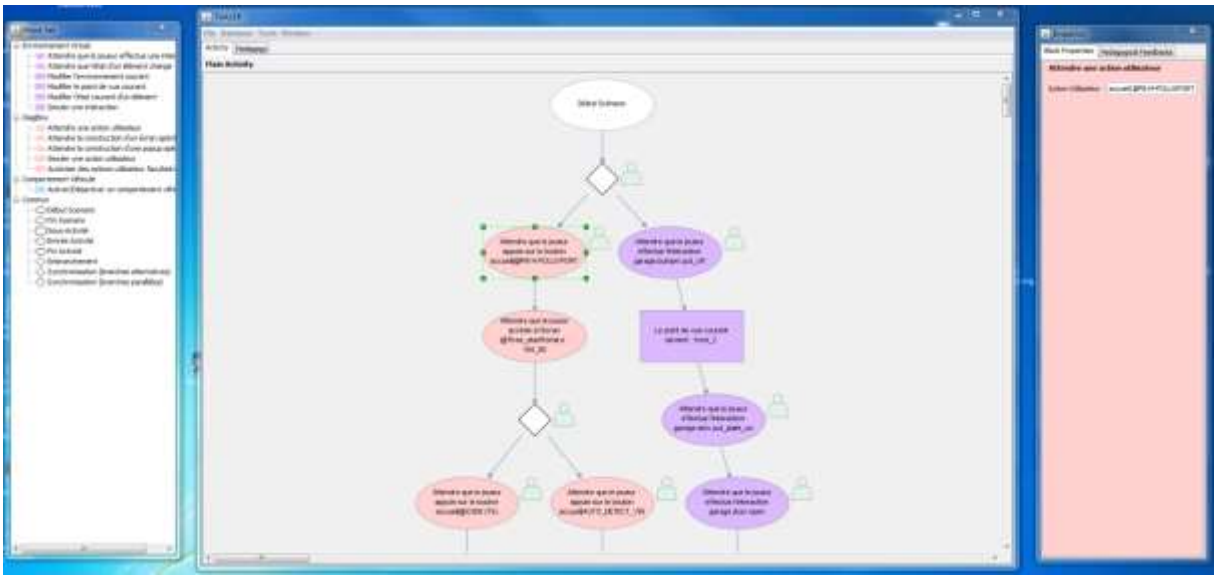


FIGURE 42. VUE PRINCIPALE LORS DE LA DESCRIPTION DE L'ACTIVITE

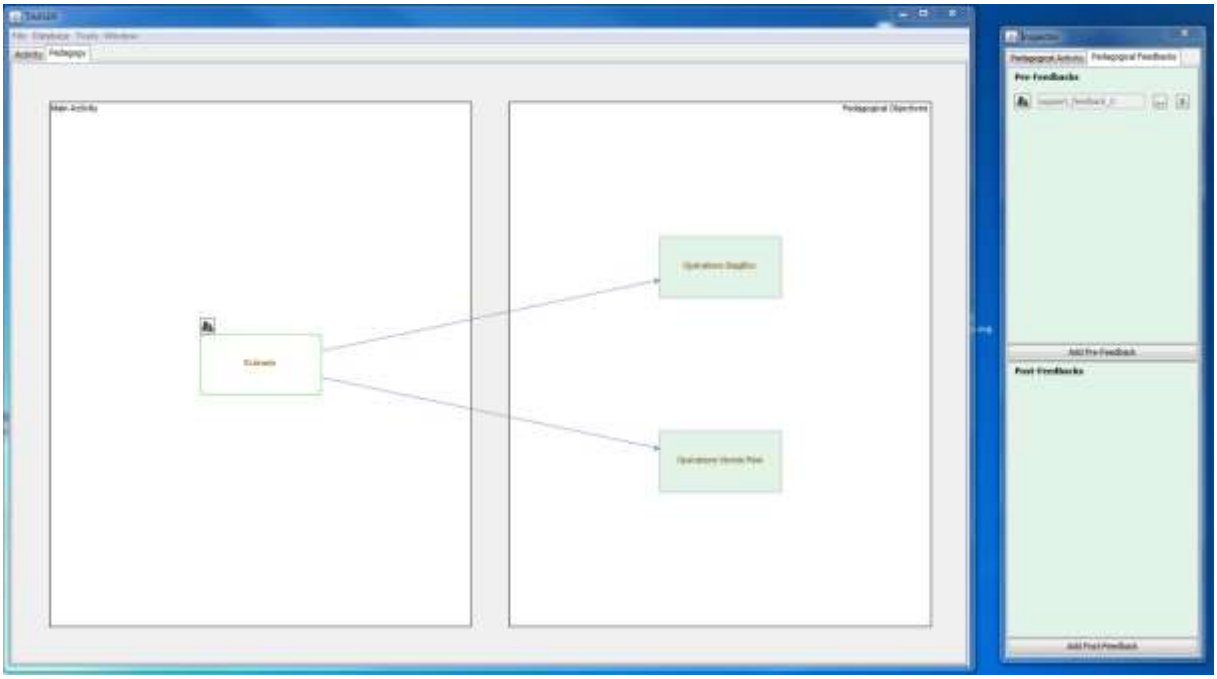


FIGURE 43. VUE PRINCIPALE LORS DE LA DESCRIPTION DE LA PEDAGOGIE

Finalement, notre environnement auteur implémente globalement la méthodologie présentée dans le Chapitre 4 afin de décrire les scénarios de formations et leurs différentes dimensions. Il présente cependant plusieurs différences :

- comme nous l'avons expliqué précédemment, tous les éléments liés à la dimension ludique de nos scénarios (mission, sous-missions, éléments de mise en scène) n'ont pas été implémentés ;
- la représentation de la hiérarchie des blocs d'activité pédagogique ne se fait plus sous forme d'arbre. A la place, nous présentons les blocs d'activité pédagogique niveau par

niveau : en commençant par le bloc d'activité pédagogique racine, l'utilisateur peut double-cliquer sur n'importe quel bloc d'activité pédagogique pour visualiser ses blocs d'activité pédagogique enfants. Dans le cas de scénarios complexes avec de multiples activités pédagogiques, cela permet de faciliter la lecture de la représentation graphique associée à la dimension pédagogique.

- l'association des dimensions s'effectue à l'aide d'un bloc particulier : le bloc de sous-activité (rangé sous la catégorie "Commun"). Ces blocs de sous-activité sont alors directement associés à un bloc d'activité pédagogique puisque notre environnement n'implémente pas la dimension ludique et les blocs de sous-mission associés. De plus, l'utilisateur doit spécifier le comportement de ce bloc en double-cliquant dessus pour accéder à son contenu. Il peut ensuite instancier normalement des blocs d'activités comme il le ferait pour décrire l'Activité. Notons alors l'apparition de deux blocs d'activités supplémentaires : le bloc d'entrée de sous-activité, et le bloc de sortie de sous-activité. Une instance au moins de chacun de ces blocs doit être créée dans la description d'un bloc de sous-activité afin d'en définir le ou les débuts et la ou les fin. C'est ce qui permet alors de lier ce bloc de sous-activité dernier à d'autres instances de blocs d'activité.
- les tags pédagogiques ne présentent maintenant plus qu'une icône dans la description de l'Activité. L'association de retours pédagogiques se fait maintenant au travers de l'inspecteur des instances de bloc d'activité. De plus, afin d'accélérer le processus de description du scénario, nous avons associé à chaque bloc d'activité les types de retour pédagogiques qui seront automatiquement créés lorsque le bloc d'activité sera instancié. Cela permet d'indiquer à l'auteur qu'il y a une forte chance qu'un retour pédagogique soit requis lors de l'utilisation d'un bloc particulier. Par exemple, le bloc "Attendre que le joueur effectue une interaction" instancie automatiquement un retour pédagogique de type *Aide* car il faut que l'auteur précise comment effectuer cette interaction au cas où l'apprenant soit bloqué à cette étape pendant l'exécution du scénario.
- les concepts de bloc composite et de base de données pédagogique/ludique globale (commune à tous les scénarios) n'ont pas été implémentés faute de temps.

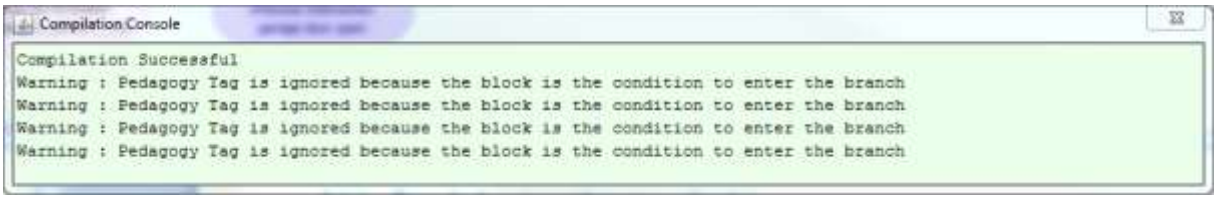


FIGURE 44. FENETRE DE DIALOGUE AFFICHANT LE RESULTAT DE LA VALIDATION DU SCENARIO

5.3.2 Base de données pédagogique (locale)

En dehors de la description du scénario lui-même, notre environnement auteur permet à l'utilisateur de gérer l'ensemble des retours pédagogiques qui sont utilisés pendant le scénario au travers de ce que nous avons appelé la base de données pédagogique. De la même manière que pour notre moteur d'exécution, les types de retours pédagogiques gérés par notre outil sont les messages d'accompagnement, les messages d'aide, et les messages d'erreur. L'utilisateur accède à

cette base de données en choisissant l'option associée dans la barre de menus en haut de l'interface principale (voir 5.3.1). Une fenêtre de dialogue s'ouvre alors (Figure 45), permettant à l'auteur :

- de créer / supprimer des retours pédagogiques dans la liste de gauche pour les deux types cités précédemment (chacun étant représenté par un onglet). L'environnement auteur définit par ailleurs des retours pédagogiques par défaut (nécessaires pour tous les scénarios). Ces messages ne sont ni modifiables, ni supprimables.
- de modifier directement dans la liste de gauche l'identifiant de chaque retour pédagogique. C'est cet identifiant qui sera référencé par les blocs de retours pédagogiques lors de la description de l'Activité.
- de modifier dans la partie droite de la fenêtre le contenu de chaque retour pédagogique. Étant donné que ces retours pédagogiques correspondent tous à des messages purement textuels, l'interface pour les modifier consiste en un simple champ de texte éditable par l'utilisateur. De plus, afin d'être en mesure de gérer plusieurs langages, notre interface propose un champ de texte par langage (dans notre cas, notre implémentation se limite à l'anglais et au français mais devra plus tard prendre en compte l'ensemble des langues proposées par DiagBox.

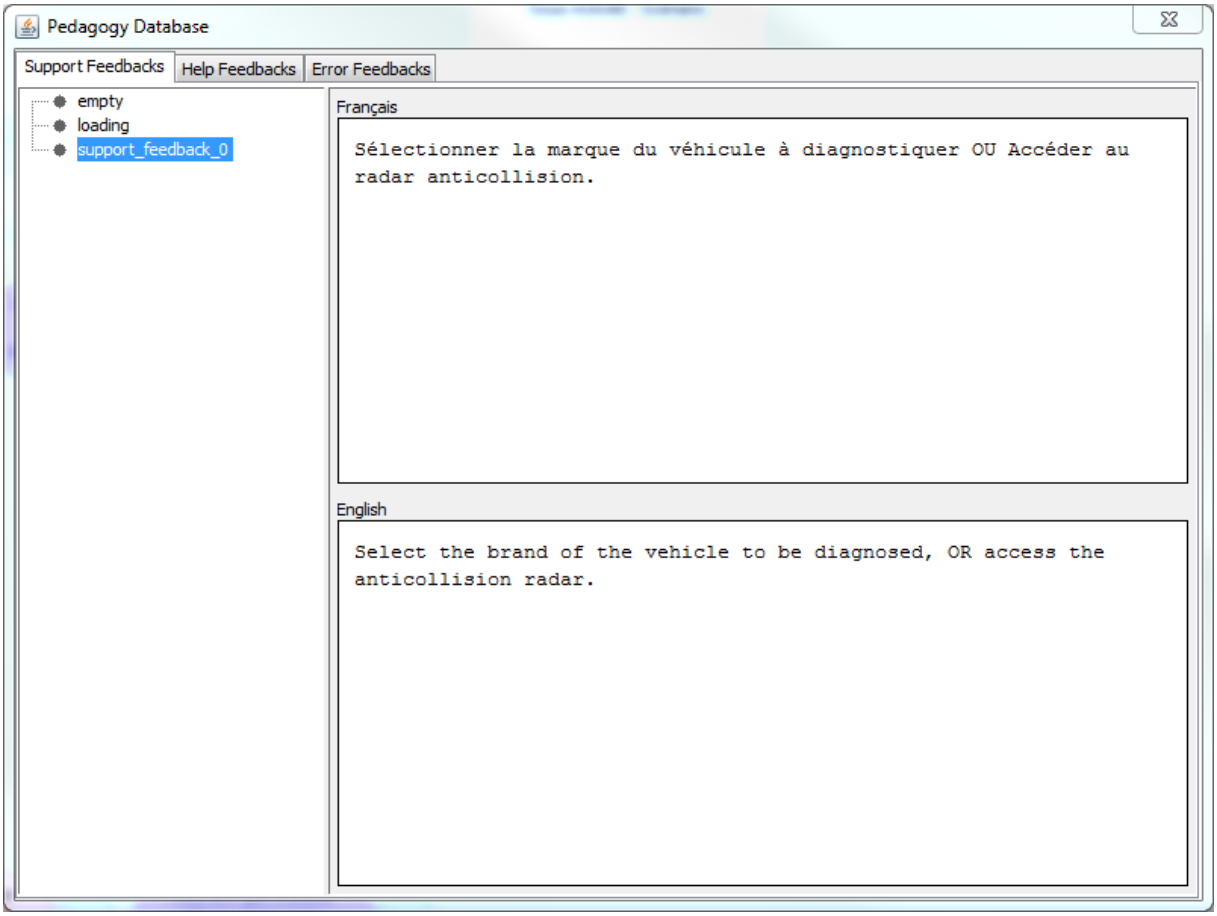


FIGURE 45. FENETRE DE DIALOGUE POUR LA GESTION DE LA BASE DE DONNEES PEDAGOGIQUE LOCALE AU SCENARIO COURANT

Finalement, l'utilisateur peut utiliser les différents éléments de cette base de données pédagogique par l'intermédiaire des fenêtres d'inspection des instances de bloc d'activité ou des blocs d'activité pédagogique. Comme présenté précédemment, ces fenêtres permettent d'associer

à ces blocs autant de retours pédagogiques que nécessaire (aussi bien en tant que pré-action que post-action). Afin de définir complètement ces retours, il est alors nécessaire de les lier avec un retour pédagogique de la base de données. Pour cela, notre inspecteur permet d'ouvrir dans une fenêtre l'onglet de la base de données qui correspond au type du retour pédagogique en cours de définition. L'utilisateur peut alors choisir celui qu'il souhaite associer, puis valider. Notons que cette fenêtre lui donne aussi la possibilité de créer/supprimer/modifier des retours pédagogiques dynamiquement afin que l'auteur n'ait pas à manipuler simultanément deux fenêtres présentant globalement les mêmes informations.

5.3.3 Gestion de l'état de l'Environnement Virtuel

Au cours de nos travaux et des différents développements effectués, nous nous sommes aperçus qu'il serait intéressant pour l'auteur de pouvoir définir l'état initial de l'Environnement de Formation au démarrage du scénario sans avoir à utiliser des blocs d'activité de type *Animation* pour placer manuellement les différents modules dans un état particulier. Pour cela, l'utilisateur peut ouvrir une fenêtre de dialogue dédiée en utilisant l'action appropriée dans la barre des menus présentée en 5.3.1. La Figure 46 présente l'interface qui est mise à disposition de l'auteur :

- la colonne A présente la liste de tous les environnements présents dans l'EV. L'environnement en gras est celui qui sera affiché au démarrage du scénario.
- la colonne B présente la liste de tous les points de vue associés à l'environnement sélectionné dans la colonne A. Le point de vue en gras représente celui dans lequel sera placé l'apprenant la première fois qu'il accèdera à l'environnement sélectionné. De plus, l'auteur peut configurer l'Environnement Virtuel pour indiquer les points de vue qui seront accessibles par l'apprenant au début du scénario, et ceux qui ne le seront pas (cette information est représentée par la coche à droite du nom du point de vue).
- La colonne C présente la liste de tous les éléments associés à l'environnement sélectionné dans la colonne A. Chaque élément peut-être :
    - rendu visible ou invisible au démarrage du scénario (icône de l'œil) ;
    - rendu interactif ou non au démarrage du scénario (icône de coche).
- La colonne D présente la liste de tous les états associés à l'élément sélectionné dans la colonne C.
- Le panneau E présente la valeur initiale de l'état sélectionné dans la colonne D. Le panneau utilise un élément d'interface différent en fonction du type de la valeur associée à l'état (case à cocher pour le type booléen, sélecteur de nombre pour les types intégraux et décimaux, et champ de texte pour le type chaine de caractères).
- La colonne F présente la liste des interactions associées à l'élément sélectionné dans la colonne C. Chaque interaction peut être rendue accessible ou non au démarrage du scénario (icône de coche).
- Le panneau G présente la liste des changements d'état associés à l'exécution de l'interaction sélectionnée dans la colonne F.

L'auteur est alors libre de configurer l'état initial de l'Environnement Virtuel comme il le souhaite en manipulant l'ensemble des informations présentées ci-dessus (notamment par l'intermédiaire de menus contextuels s'affichant après un clic droit dans la colonne adaptée).

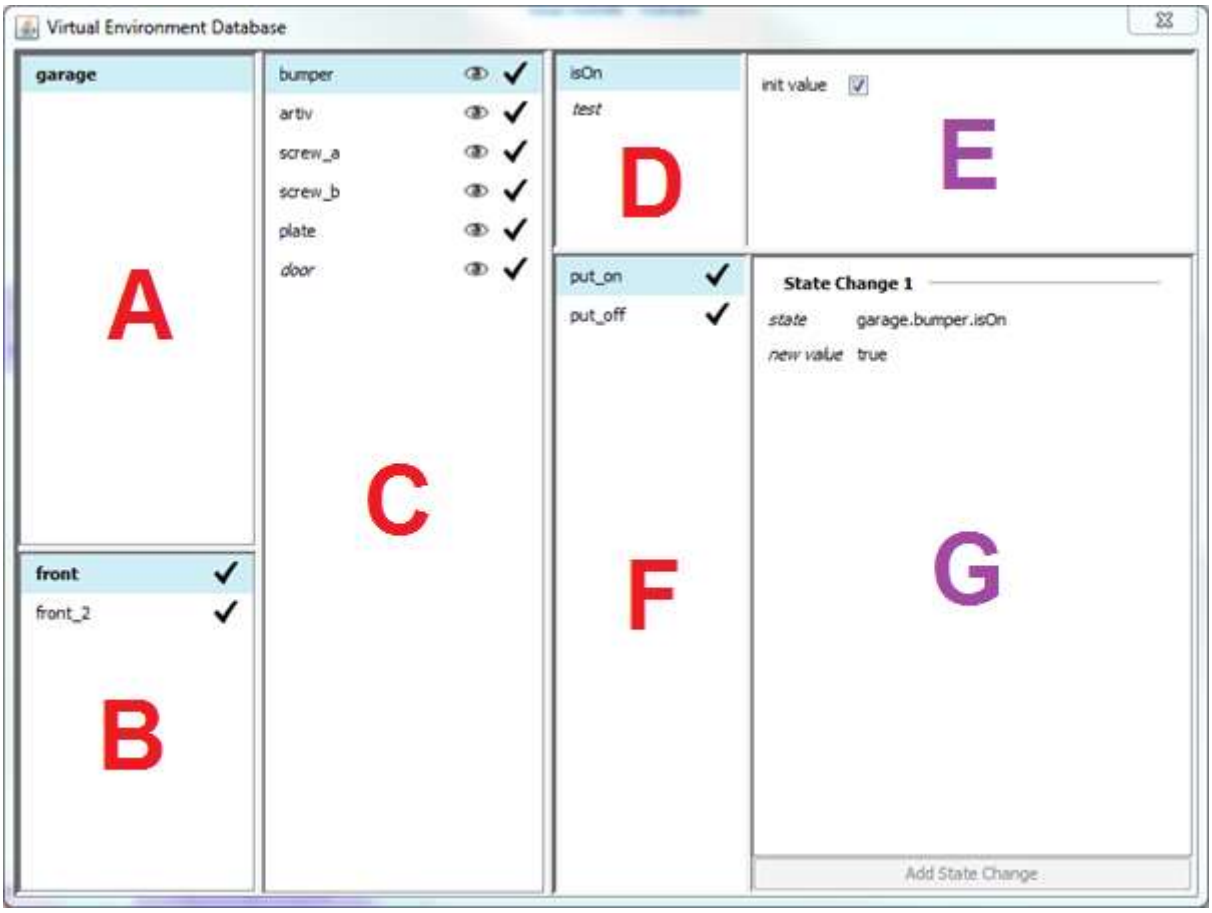


FIGURE 46. FENETRE DE DIALOGUE POUR LA GESTION DE L'ETAT INITIAL DE L'ENVIRONNEMENT VIRTUEL

Une deuxième problématique liée à la gestion de l'Environnement Virtuel est associée au fait que l'auteur ne puisse décrire des opérations s'effectuant dans ce module que par rapport aux objets du modèle qui existent effectivement dans la version actuelle de l'EV. Si par exemple, pour les besoins d'un nouveau scénario, l'auteur souhaite décrire une interaction sur un élément qui n'existe pas dans le modèle actuel, il se retrouve alors dans l'impasse et doit demander à l'équipe de développement dédiée d'enrichir l'EV avant qu'il ne puisse décrire son scénario. Sans demander à l'auteur d'effectuer lui-même ces développements qui requièrent des compétences avancées en informatique, nous avons souhaité lui donner la possibilité de décrire des objets factices afin qu'il puisse malgré tout créer et tester son scénario.

Pour cela, la fenêtre de dialogue présentée dans la Figure 46 permet à l'auteur (toujours via l'utilisation de menus contextuels après un clic droit dans les colonnes adaptées) :

- de décrire des environnements factices. L'auteur doit alors simplement en fournir l'identifiant.
- de décrire des points de vue factices (pour chaque environnement, factice ou non). L'auteur doit alors simplement en fournir l'identifiant.
- de créer des éléments factices (pour chaque environnement, factice ou non). L'auteur doit alors simplement en fournir l'identifiant.

- de créer des états factices (pour chaque élément, factice ou non). L'auteur doit alors en fournir l'identifiant et le type.
- de créer des interactions factices (pour chaque élément, factice ou non). L'auteur doit alors simplement en fournir l'identifiant.
- de créer des changements d'états, pour les interactions factices uniquement (nous avons en effet choisi de ne pas pouvoir altérer les interactions "existantes" afin de ne pas en changer le sens intrinsèque). Pour créer ces changements d'états, l'auteur doit :
    - identifier l'état à modifier (via des menus déroulants)
    - spécifier la valeur que doit prendre cet état (de manière similaire au panneau E, l'élément d'interface permettant de spécifier cette valeur dépend du type de la valeur).

Par la suite, l'exécution de l'Environnement Virtuel par l'intermédiaire de notre environnement auteur et de l'interface informatique mise à disposition par l'EV (voir 5.2.1.1) permet d'instancier tous ces objets factices. La Figure 47 montre la présentation de ces objets sous la forme de menus déroulants. L'auteur est alors en mesure d'enregistrer des séquences d'actions en utilisant à la fois les objets existants et les objets factices.

FIGURE 47. CAPTURE D'ECRAN DE L'ENVIRONNEMENT VIRTUEL DANS LEQUEL L'ELEMENT FACTICE "PORTE" A ETE AJOUTE, AVEC LES DEUX INTERACTIONS FACTICES "OUVRIR" ET "FERMER"

Finalement, l'opération d'exportation du scénario permettant aussi d'exporter ces objets factices, l'auteur sera aussi capable de tester son scénario au travers de notre moteur d'exécution bien que certains objets n'aient pas de réelle représentation visuelle à l'exécution de

l'Environnement Virtuel. Nous proposons dans nos Perspectives des idées complémentaires pour améliorer le processus global de création de scénarios et d'enrichissement de l'Environnement Virtuel associé.

5.3.4 Évaluation

Au terme de l'implémentation de cette version prototype de l'environnement auteur, nous avons souhaité évaluer sa pertinence. Pour cela, notre objectif initial fut logiquement de faire tester notre environnement par les personnes ciblées par la méthodologie proposée dans le Chapitre 4 : les formateurs. En effet, il n'y aurait eu que très peu de sens à le faire tester par des personnes ne possédant ni l'expertise métier, ni l'expertise pédagogique, qui constituent finalement le point de départ de notre méthodologie. Nous avons ainsi défini un certain nombre de métriques nous permettant d'évaluer différents aspects de notre environnement auteur :

- impressions générales ;
- comparaison entre le temps requis pour décrire un scénario manuellement (en utilisant directement les formats introduits dans le 0) et par l'intermédiaire de notre environnement auteur ;
- ergonomie et prise en main ;
- capacité pour les formateurs à utiliser notre représentation graphique pour décrire un nouveau scénario ;
- capacité pour les formateurs à lire, comprendre et modifier un scénario existant.

Cependant, de par des contraintes humaines (demande d'autorisation auprès de PSA, disponibilité des formateurs) et géographiques (déplacement sur Paris nécessaire), il ne nous a pas été possible d'organiser cette phase d'évaluation auprès des formateurs dans les délais impartis. Afin malgré tout d'être en mesure de juger à minima la pertinence ou non de notre environnement auteur, nous avons choisi de réaliser une phase de test en internes à ACTIA afin de d'évaluer la première métrique.

5 personnes se sont alors prêtées au jeu, toutes possédant un solide bagage en programmation afin de se positionner dans un cas extrême (les formateurs ne possédant pas nécessairement ce bagage). De plus, afin de pallier au manque d'expertise métier et pédagogique, chaque personne avait à sa disposition la fiche de la leçon à modéliser de manière similaire à l'exemple donné dans la partie 3.3.2 (le scénario en question étant le réglage du radar anticollision, décrit en Annexe 1). Au terme de cette évaluation, nous avons obtenu les résultats suivants :

- en moyenne 2 jours de travail ont été nécessaires afin que les participants obtiennent une version fonctionnelle du scénario de manière manuelle ;
- en moyenne une demi-journée de travail a été nécessaire afin que les participants obtiennent une version fonctionnelle du scénario à l'aide de l'environnement auteur.

Cela met en évidence le fait que l'utilisation de notre environnement auteur accélère de manière non-négligeable le processus de création des scénarios pour des personnes étant familières avec des concepts informatiques avancés. Après avoir échangé avec les testeurs, cette différence s'explique principalement par le fait qu'il soit nécessaire de connaître les identifiants internes de chaque écran et de chaque bouton dans le logiciel de diagnostic, et de connaître la syntaxe permettant d'identifier chaque propriété de l'Environnement de Formation dans les instructions de `Checkpoint` et d'`Animation`. Dès lors, nous pouvons raisonnablement supposer

que le constat serait le même voire encore plus marqué en considérant une population de test composée de formateurs. Cela nous amène à conclure sur l'efficacité certaine de notre environnement auteur vis-à-vis d'un processus de création entièrement manuel.

Cependant, comme expliqué plus haut, cette évaluation ne nous permet pas de conclure sur la pertinence des différentes représentations graphiques proposées, et plus généralement sur la pertinence de notre méthodologie. Ainsi, pour mieux donner une idée du résultat final, nous donnons au lecteur un lien vers une vidéo disponible en ligne[42] faisant une démonstration de notre environnement auteur.

5.4 Synthèse

Ce chapitre nous a permis de présenter l'outil de formation implémenté lors de cette thèse ainsi que l'environnement auteur associé permettant de rendre la création de scénarios accessible aux formateurs. L'ensemble des développements ainsi effectués s'appuient sur les différents concepts introduits au travers du 0 et du Chapitre 4 de ce mémoire, et les met en application dans le contexte industriel particulier du diagnostic automobile.

Nous avons ensuite procédé à plusieurs évaluations afin de mesurer la pertinence de ces deux systèmes :

- En ce qui concerne l'outil de formation, deux phases de test ont été menées successivement afin de proposer à des garagistes d'en évaluer plusieurs critères. Finalement, les réponses obtenues via les questionnaires fournis nous ont amenés à valider la pertinence générale de notre outil de formation.
- En ce qui concerne l'environnement auteur, nous avons défini plusieurs grandes métriques permettant d'en évaluer les différents aspects. Cependant, l'accès au formateur n'étant pas évident du fait qu'il fasse partie d'un autre organisme, nous n'avons pas pu mener cette phase dans les délais impartis. Nous avons malgré tout choisi de tester une partie des métriques concernées en interne, en comparant le temps requis pour créer un scénario à la main d'une part, et via notre environnement auteur d'autre part. En se basant sur le scenario présenté en Annexe 1, nous avons alors conclu de l'efficacité de notre environnement. Puis, afin que le lecteur puisse obtenir une meilleure idée du résultat finale, nous lui avons indiqué un lien vers une vidéo mettant en scène notre environnement auteur autour de la description de ce même scénario.

Finalement, en se restreignant à notre contexte industriel bien spécifique, les différentes évaluations effectuées nous amènent à valider dans l'ensemble les travaux présentés dans ce mémoire.

[42] https://1drv.ms/f/s!AhGc1jP4zCwFhqx8MhhGd6cnl40o-w. Une première vidéo présente la description d'un scenario similaire à celui présenté en 3.3.2, et une deuxième vidéo montre l'exécution de ce scénario au travers de l'outil de formation implémenté.

Conclusion et Perspectives

Dans cette dernière section de notre manuscrit, nous synthétisons tout d'abord les raisons nous ayant conduit à étudier les Jeux Sérieux et les tutoriels de logiciels, ainsi que différentes méthodes et environnements permettant la création assistée de tout ou partie de ces outils de formation. Puis, nous récapitulons les travaux que nous avons accomplis durant cette thèse, avec les différentes contributions apportées vis-à-vis de notre état de l'art et leur application dans le domaine industriel précis du diagnostic automobile. Nous terminons cette ultime partie par la présentation des travaux sur lesquels il serait intéressant de se pencher afin de compléter et d'améliorer ceux présentés dans ce mémoire.

Synthèse

Motivations

Les différentes études menées pendant cette thèse ont dérivé du constat initial purement industriel que les véhicules d'aujourd'hui comportaient de plus en plus de composants électroniques, et que la difficulté à diagnostiquer et à réparer des pannes sur ces véhicules augmentait en conséquence. Dans un premier temps, nous nous sommes alors intéressés à explorer différents types d'outils de formation dont l'objectif principal serait de combler l'accroissement constant de la complexité des procédures ou activités métier associées. Les solutions à cette problématique étant multiples, nous nous sommes alors appuyés sur deux critères principaux afin de préciser nos recherches :

- les activités métier qui nous intéressent impliquent l'interaction avec des objets du monde réel (ex : la voiture) et avec un ou plusieurs logiciels associés (ex : le logiciel de diagnostic) ;
- afin d'augmenter l'intérêt inhérent porté par les apprenants aux formations, nous avons souhaité adopter un processus d'apprentissage actif les impliquant directement, notamment via l'utilisation de ressors ludiques.

Notre intérêt s'est alors naturellement porté vers l'analyse des Jeux Sérieux, une forme d'outil de formation combinant par définition les dimensions pédagogiques et ludiques de manière homogène (1.2.1.1). Puis, afin d'intégrer directement dans la formation et de manière cohérente le ou les logiciels métiers associés aux activités étudiées (1.3), nous avons ensuite étendu notre étude à l'analyse des différentes formes de tutoriels de logiciel existantes.

A la suite de cette étude, un deuxième constat fut alors d'établir le fait que le développement de tels outils de formation nécessitait des compétences informatiques pointues, et n'était donc pas à la portée de la majorité des personnes devant être fortement impliquées dans la conception des sessions de formation : les formateurs. Dans un second temps, nous nous sommes alors intéressés à l'étude de différentes méthodes et environnements permettant de rendre le développement de certaines parties de ces outils de formation accessibles à des personnes sans compétence informatique particulière. Ainsi, nous avons dirigé notre état de l'art en suivant deux axes principaux :

- l'analyse de différents formats de représentations textuels ou graphiques permettant de modéliser des dimensions particulières de différents outils de formation. Nous avons alors pu mettre en évidence des concepts et/ou des notations adaptés aux compétences du formateur, tels que BPMN ou la notion de blocs graphiques configurables.
- l'analyse de différents outils et environnements permettant d'assister les formateurs dans la conception et le développement de ces outils de formation. Cela nous a alors permis de faire ressortir plusieurs méthodes de génération automatisée, ainsi que les caractéristiques importantes des environnements auteur visant à faciliter la création de contenu pour différents types d'application.

Finalement, les différentes études menées dans les deux premiers chapitres de ce mémoire nous ont servi de point de départ afin de répondre aux deux problématiques principales de nos travaux :

1. Concevoir un outil de formation pour l'enseignement d'activités métier dans un environnement informatique modulaire ;
2. Définir une méthodologie adaptée aux formateurs leur permettant de créer leurs propres scénarios de formation.

Contributions

La première grande contribution que nous ayons apportée au cours de nos travaux est liée à la définition de notre outil de formation et de ses deux composantes principales :

- l'Environnement de Formation. Il est constitué de l'ensemble des modules nécessaires à la simulation de l'environnement de travail habituel de l'apprenant pour les activités métier étudiées :
    - l'Environnement Virtuel, qui permet de simuler les objets du monde réel avec lesquels doit interagir l'apprenant ;
    - les Logiciels Métier, qui représentent les logiciels réellement utilisés afin d'accomplir l'activité étudiée ;
    - les Systèmes de Simulation, qui servent à simuler les échanges entre le monde réel les Logiciels Métier afin que ces derniers fonctionnent correctement (en regard avec l'activité étudiée).
    - le Contexte Partagé, qui est en charge d'interfacer l'EV, les LM et les SS afin que des environnements extérieurs puissent interagir de manière homogène avec l'ensemble des modules de l'EF.
- le Scénario. Il s'agit de l'élément de notre outil qui permet de représenter formellement une leçon en s'appuyant sur le modèle de l'Environnement de Formation ainsi que sur une base de données d'éléments pédagogiques et ludiques. Pour cela, nous avons d'un côté défini les différentes composantes des dimensions pédagogiques et ludiques qu'il nous a semblé pertinent d'intégrer à notre outil de formation, ainsi que le format permettant de représenter ces composantes. De l'autre côté, nous avons introduit notre propre langage dédié permettant de décrire à la fois l'évolution de l'EF en fonction des actions de l'apprenant et les éléments pédagogiques et ludiques liées à la leçon étudiée.

Notre deuxième grande contribution concerne la définition d'une méthodologie devant permettre aux formateurs de créer leurs propres scénarios au travers d'un environnement auteur. En effet, le langage dédié alors introduit étant par nature destiné à être utilisé par des personnes possédant des compétences en programmation, il était difficilement imaginable de le mettre directement à disposition des formateurs. Cependant, du fait que ce soit eux qui possèdent l'expertise pédagogique et métier requise pour décrire des scénarios de manière pertinente, nous nous sommes intéressés à définir une méthodologie qui soit adaptée à leur profil. Ainsi, nous avons dans un premier temps proposé un langage graphique qui permette de représenter séparément les différentes dimensions de nos scénarios afin que l'auteur puisse les traiter une par une sans se préoccuper des autres. Nous avons ensuite introduit un élément permettant d'associer entre elles chacune de ces dimensions par l'intermédiaire de la description de l'Activité. Enfin, nous avons décrit le processus devant être implémenté par l'environnement auteur afin de convertir la représentation graphique finale du scénario vers son équivalent dans le langage dédié.

Dans un second temps, nous avons introduit différents éléments visant à améliorer et à accélérer le processus global de description des scénarios. Pour cela, nous avons d'abord défini deux fonctionnalités proposant d'intégrer directement l'Environnement de Formation dans le processus :

- l'environnement auteur proposera à l'auteur la possibilité d'enregistrer des séquences d'actions en interagissant directement dans les différents modules de l'EF pour ensuite générer automatiquement la représentation graphique correspondante.
- l'environnement auteur permettra de vérifier la cohérence de la description de l'Activité, en particulier en ce qui concerne la description d'activités alternatives et/ou parallèles. L'objectif final est de s'assurer que l'ensemble du scénario est jouable et cohérent via la modification automatique de l'EF.

Nous avons ensuite proposé deux nouvelles fonctionnalités permettant de réutiliser plusieurs éléments de scénarios déjà décrits. Ainsi, nous proposons d'abord le principe des blocs composites qui permettent de grouper des séquences d'instances de bloc d'activité dans un unique bloc. Ce bloc sera alors intégré à la bibliothèque de blocs disponible pour la description de l'Activité de futurs scénarios. Puis, nous avons introduit une méthode visant à permettre à l'auteur d'exporter des retours pédagogiques et des éléments de mise en scène pour qu'ils puissent être réutilisés tel quel ou de manière adaptée pour la description de la pédagogie et des éléments ludiques de futurs scénarios.

Finalement, nous avons décrit notre implémentation de l'outil de formation mettant en application la première partie de nos travaux dans le contexte industriel du diagnostic automobile. Grâce à deux phases d'évaluation successives dans plusieurs garages, nous avons alors été en mesure de valider la pertinence de notre approche auprès des garagistes cibles, en fonction de nos travaux. De la même manière, nous avons présenté notre implémentation de l'environnement auteur associé à l'outil de formation décrit. Bien que toutes les fonctionnalités introduites dans le Chapitre 4 n'aient pas été intégrées dans notre prototype, nous avons pu vérifier que son utilisation était plus efficace que la manipulation directe de notre langage dédié afin de décrire des scénarios. Cependant, du fait qu'une véritable phase d'évaluation auprès de la population cible (les formateurs) n'ait pas pu être menée, nous avons simplement proposer au lecteur un lien vers une vidéo afin qu'il puisse avoir une meilleure idée du résultat final.

Perspectives

Réalisations et Évaluation

Du fait que certains besoins de notre client aient guidé les développements effectués durant cette thèse, mais aussi du fait que nous souhaitions obtenir des prototypes fonctionnels dans les délais imposés, les points abordés dans le 0 et le Chapitre 4 n'ont pas tous été appliqués dans nos réalisations. En particulier, les points suivants font partie de ceux à approfondir en priorité :

- ***Gestion des éléments ludiques***. Ni l'outil de formation ni l'environnement auteur implémenté ne gère les différents éléments ludiques introduits dans la partie 3.2.3.2 afin d'apporter à l'apprenant une motivation supplémentaire de suivre la formation. Nous prévoyons donc d'intégrer la gestion de ces éléments dans nos deux systèmes. Dans un premier temps, nous souhaitons introduire le principe de mission et de sous-missions, avec tous les éléments qui leur sont directement liés : difficulté, score requis pour valider le scénario, carnet de bord. Dans un second temps, nous inclurons plusieurs types d'élément permettant de mettre en scène le scénario : dialogues, vidéos, etc.
- ***Enrichir la dimension pédagogique***. Actuellement, notre outil de formation ne gère que trois types de retour pédagogique : accompagnement, aide, et erreur. Nous prévoyons donc d'en ajouter de nouveaux fin d'être en mesure d'apporter des retours pédagogiques plus précis et plus pertinent en fonction de la situation (ex : conseils afin d'apporter des informations complémentaires en rapport avec les actions devant être effectuées par le joueur, questionnaires afin de vérifier les connaissances de l'apprenant, etc.). De plus, il nous faut intégrer le système de récapitulation de la leçon, afin de pouvoir faire un retour général à la fin de la leçon sur les performances de l'apprenant. Bien entendu, notre environnement auteur devra prendre en considération ces modifications afin que l'auteur puisse exploiter ces nouveaux éléments lors de la description de ses scénarios. Nous souhaitons faire intervenir des formateurs afin de recueillir leurs idées et d'introduire ces nouveaux éléments de la manière la plus pertinente possible.
- ***Gestion des bases de données locales et globales***. Nous souhaitons implémenter le principe de base de données locale et globale introduit dans la partie 4.3.2 afin de pouvoir capitaliser des séquences de bloc d'activité (par l'intermédiaire des blocs composites), des retours pédagogiques, et des éléments de mise en scène. La mise en place d'un tel système pose certaines problématiques, en particulier en ce qui concerne la gestion de la base de données globale et les accès concurrents (pour modifier des éléments par exemple). Nous envisageons alors un fonctionnement global similaire à celui utilisé par le gestionnaire de configuration git[43] : chaque utilisateur possède une copie de la base de données globale (disponible sur un serveur) sur son poste de travail, et peut la modifier comme il le souhaite. Il peut ensuite soumettre ces modifications au serveur, qui devront alors être validées par une personne tierce (à définir) avant d'être acceptées.
- ***Assurer la cohérence de la description de l'Activité***. Nous prévoyons d'implémenter au plus tôt dans notre environnement auteur les différents processus permettant de s'assurer que l'ensemble des scénarios décrits sont entièrement jouables (4.3.1.2).

[43] https://git-scm.com/

- ***Introduire la génération des scripts Spy And Sim et des Comportements associés dans le processus global.*** Lors de la description de scénarios par le biais de notre environnement auteur, nous avons posé l'hypothèse que l'utilisateur avait à sa disposition l'ensemble des scripts Spy And Sim et les Comportements (3.2.2.3) associés. Dans les faits, cette hypothèse est rarement vérifiée puisque chaque scénario impliquera souvent d'étudier une fonctionnalité particulière d'un véhicule particulier, et éventuellement d'introduire des éléments de fautes afin de simuler des mauvais fonctionnements. Un script Spy And Sim devra alors presque toujours être créé spécifiquement pour chaque nouveau scénario (il sera éventuellement possible de reprendre un script existant afin de l'adapter), ne rentrant pas dans les compétences du formateur et impliquant donc l'intervention d'un expert. Nous souhaitons donc réfléchir à un système qui permette au formateur, via l'environnement auteur, de générer et personnaliser ses propres scripts Spy And Sim (et les Comportements associés) de façon adaptée à son expertise.
- ***Évaluation de l'environnement auteur et de la méthodologie associée***. Afin de compléter l'évaluation menée dans la partie 5.3.4, nous souhaitons mettre notre environnement auteur à disposition des formateurs. L'objectif sera alors de recueillir leurs impressions sur la méthodologie globale, et en particulier sur la représentation graphique associée à chaque dimension, afin de pouvoir ensuite continuer à améliorer le processus global et à l'adapter aux compétences des formateurs.

De plus, nous avons identifié certaines fonctionnalités spécifiques à l'Environnement Virtuel qu'il serait intéressant d'implémenter afin de renforcer son côté modulaire et de pouvoir en profiter pleinement dans le processus global de description des scénarios :

- ***Génération d'un cahier des charges***. L'environnement auteur permet actuellement de décrire un ensemble d'objets factices à ajouter au modèle de l'EV afin de décrire des situations et des interactions qui ne sont pas implémentées dans la version courante de l'EV. Ces objets sont ensuite gérés de manière générique sous forme de menus déroulant qui se succèdent. Nous envisageons alors d'intégrer à notre environnement auteur une fonctionnalité permettant de générer automatiquement un cahier des charges qui décrivent ces objets factices à partir du modèle décrit par le formateur. Des informations complémentaires pourront être apportées de manière purement textuelle afin par exemple de donner des indications sur la représentation graphique de tel élément en fonction de l'état global de l'EV. Les personnes qui seront ensuite en charge d'implémenter ces modifications pourront alors simplement suivre l'ensemble des informations apportées par ce cahier des charges. Les scénarios décrits par le formateur seront ainsi toujours exécutables.
- ***Script pour décrire des comportements avancés***. Actuellement, il est possible d'adapter certains éléments de l'Environnement Virtuel (afin qu'il corresponde à des besoins spécifiques d'un nouveau scénario) sans avoir à modifier le code source directement. Comme décrit en 5.2.1.1, il ainsi possible de modifier ou de définir des nouveaux environnements, éléments, interactions, … et certaines de leurs propriétés dans les fichiers correspondants. Il n'est cependant pas possible de décrire des comportements spécifiques, tel que les changements de position et/ou d'image d'un élément en fonction de l'état global de l'EV. Afin de parvenir à une solution complète et générique, nous avons songé à définir ces comportements "complexes" dans des fichiers de script qui seraient ensuite exécutés par un moteur spécifique intégré à notre Environnement

Virtuel. Pour cela, une piste envisagée est l'intégration d'un module Lua[44] dans notre EV, Lua étant un langage de script très largement utilisé dans le domaine du Jeu Vidéo et, par continuité, dans le domaine des Jeux Sérieux.

- ***Enrichir le modèle***. Le modèle de l'Environnement Virtuel actuel permet de décrire des scènes 2D/3D relativement statiques. Nous envisageons d'enrichir ce modèle afin de pouvoir ajouter plus de dynamisme lors de son utilisation, mais aussi afin tout simplement de pouvoir modéliser un panel de situations plus large. Des pistes pour cela concernent l'ajout d'autres types d'interaction (ex : rotation de la souris pour faire tourner un élément, tel qu'un tournevis) ou une gestion plus précise des changements d'état (ajout de condition, opérations plus complexes pour définir la nouvelle valeur de l'état, …). De plus, pour les phases très spécifiques pouvant difficilement être décrite en utilisant notre modèle générique (ex : phase de roulage du scénario décrit en Annexe 1), nous souhaitons pouvoir exécuter depuis notre EV des applications externes qui auront été implémentées en fonction des besoins spécifiques de la leçon associée.

Prolongement des contributions

Finalement, nous prévoyons de nous intéresser à différents sujets qui contribueraient à enrichir nos contributions initiales :

- ***Scénarios libres***. Dans la partie 3.3, nous avons introduit les deux principaux types de scénarios qu'il est possible d'imaginer et éventuellement de mixer : les scénarios guidés et les scénarios libres. Dans la suite du mémoire, nous avons choisi de nous concentrer sur la définition d'un langage et de différents processus adaptés à la description des scénarios guidés. Nous souhaitons maintenant réfléchir aux problématiques qui se posent si l'on considère des scénarios libres, et voir s'il est possible d'adapter notre méthodologie à la description de ce deuxième type de scénario.
- ***Règles de comportement***. Un autre axe permettant d'approfondir nos travaux consisterait à définir un système permettant de définir des règles de comportement. Le rôle d'un tel système serait de surveiller les changements de valeur des propriétés associées à notre Environnement de Formation, et d'y réagir indépendamment du scénario. Un exemple de règle pourrait être : "Si le comodo du feu de croisement est sur la position Allumé, les phares doivent s'allumer". Le formateur pourrait ainsi spécifier un ensemble de règles décrivant un comportement simplifié du véhicule, ou plus généralement de l'Environnement de Formation. Chacune de ces règles pourraient ensuite être activées ou désactivées pendant la progression de l'apprenant dans le scénario, afin par exemple de simuler des comportements nominaux du véhicule ou encore de reproduire des pannes.

[44] https://www.lua.org/

Bibliographie

Alvarez, J. (2007). Du jeu vidéo au serious game : approches culturelle, pragmatique et formelle. *Thèse de doctorat. Toulouse 2*.

Amato, E. A. (2007). Vers une instrumentalisation communicationnelle des jeux vidéo: quelles formes de séduction idéologique ou publicitaire ? *Colloque international EUTIC 2007 : " Enjeux et Usages des TIC"*.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological review, 89*(4), 369.

Annetta, L. A. (2010). The “I's” have it: A framework for serious educational game design. *Review of General Psychology, 14*(2), 105.

Annetta, L. A., Minogue, J., Holmes, S. Y., & Cheng, M. T. (2009). Investigating the impact of video games on high school students’ engagement and learning about genetics. *Computers & Education, 53*(1), 74-85.

Araújo, M., & Roque, L. (2009). Modeling games with petri nets. *Breaking New Ground: Innovation in Games, Play, Practice and Theory, DIGRA2009, Londres, Royaume-Uni*.

Army, U. S. (2003). America's Army: Special Forces. Vers. PC 2.0 a. *Computer software. US Army*.

Barot, C., Lourdeaux, D., Burkhardt, J. M., Amokrane, K., & Lenne, D. (2013). V3S: A virtual environment for risk-management training based on human-activity models. *Presence: Teleoperators and Virtual Environments, 22*(1), 1-19.

Benware, C. A., & Deci, E. L. (1984). Quality of learning with an active versus passive motivational set. *American Educational Research Journal, 21*(4), 755-765.

Bergman, L., Castelli, V., Lau, T., & Oblinger, D. (2005). DocWizards: a system for authoring follow-me documentation wizards. *Proceedings of the 18th annual ACM symposium on User interface software and technology, ACM*, 191-200.

Carroll, J. M. (1990). *The Nurnberg funnel: designing minimalist instruction for practical computer skill.* Cambridge: MA: MIT press.

Carroll, J. M., & Rosson, M. B. (1987). Paradox of the active user. *The MIT Press*.

Chen, H., Wigand, R. T., & Nilan, M. S. (1999). Optimal experience of web activities. *Computers in human behavior, 15*(5), 585-608.

Chi, P. Y., Ahn, S., Ren, A., Dontcheva, M., Li, W., & Hartmann, B. (2012). MixT: automatic generation of step-by-step mixed media tutorials. *Proceedings of the 25th annual ACM symposium on User interface software and technology, ACM*, 93-102.

Contreras, J., & Saiz, F. (1996). A Framework for the Automatic Generation of Software Tutoring. *CADUI*, 171-182.

Cordova, D. I., & Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of educational psychology, 88*(4), 715.

Csikszentmihalyi, M. (1988). The flow experience and its significance for human psychology.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining gamification. *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments, ACM*, 9-15.

Deterding, S., Sicart, M., Nacke, L., O'Hara, K., & Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. *CHI'11 Extended Abstracts on Human Factors in Computing Systems, ACM*, 2425-2428.

Devillers, F., & Donikian, S. (2003). A scenario language to orchestrate virtual world evolution. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association*, 365-275.

Djaouti, D. (2011). Serious Game Design : considérations théoriques et techniques sur la création de jeux vidéo à vocation utilitaire. *Thèse de doctorat. Université de Toulouse, Université Toulouse III-Paul Sabatier*.

Dong, T., Dontcheva, M., Joseph, D., Karahalios, K., Newman, M., & Ackerman, M. (2012). Discovery-based games for learning software. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM*, 2083-2086.

Duval, Y., Panzoli, D., Reymonet, A., Plantec, J.-Y., Thomas, J., & Jessel, J.-P. (2015). Serious Games Scenario Modeling for Non-Experts. *Proceedings of the 7th International Conference on Computer Supported Education (CSEDU)*, 474-479.

Farkas, D. K., & Williams, T. R. (1990). John Carroll's the Nurnberg funnel and minimalist documentation. *IEEE Transactions on professional communication, 33*(4), 182-187.

Fernquist, J., Grossman, T., & Fitzmaurice, G. (2011). Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning. *Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM*, 373-382.

Ferraris, C., Lejeune, A., Vignollet, L., & David, J. P. (2005). Modélisation de scénarios pédagogiques collaboratifs.

Gagne, R. M. (1984). Learning outcomes and their effects: Useful categories of human performance. *American Psychologist, 39*(4), 377.

Garcia, F. (2000). CACTUS: Automated tutorial course generation for software applications. *Proceedings of the 5th international conference on Intelligent user interfaces, ACM*, 113-120.

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & gaming, 33*(4), 441-467.

Girard, C., Ecalle, J., & Magnan, A. (2013). Serious games as new educational tools: how effective are they? A meta-analysis of recent studies. *Journal of Computer Assisted Learning, 29*(3), 207-219.

Grabler, F., Agrawala, M., Li, W., Dontcheva, M., & Igarashi, T. (2009). Generating photo manipulation tutorials by demonstration. *ACM Transactions on Graphics (TOG), 28*(3), 66.

Grossman, T., & Fitzmaurice, G. (2010). ToolClips: an investigation of contextual video assistance for functionality understanding. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM*, 1515-1524.

Grossman, T., Fitzmaurice, G., & Attar, R. (2009). A survey of software learnability: metrics, methodologies and guidelines. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM*, 649-658.

Habgood, M. P., Ainsworth, S. E., & Benford, S. (2005). Endogenous fantasy and learning in digital games. *Simulation & Gaming, 36*(4), 483-498.

Hainey, T., Connolly, T. M., Stansfield, M., & Boyle, E. A. (2011). Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level. *Computers & Education, 56*(1), 21-35.

Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work?--a literature review of empirical studies on gamification. *2014 47th Hawaii International Conference on System Sciences, IEEE*, 3025-3034.

Harms, K. J., Cosgrove, D., Gray, S., & Kelleher, C. (2013). Automatically generating tutorials to enable middle school children to learn programming independently. *Proceedings of the 12th International Conference on Interaction Design and Children, ACM*, 11-19.

Harteveld, C., Guimarães, R., Mayer, I., & Bidarra, R. (2007). Balancing pedagogy, game and reality components within a unique serious game for training levee inspection. *International Conference on Technologies for E-Learning and Digital Entertainment, Springer Berlin Heidelberg*, 128-139.

Ishida, T. (2002). Q: A scenario description language for interactive agents. *Computer, 35*(11), 42-47.

John, B. E., & Kieras, D. E. (1996). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction (TOCHI), 3*(4), 287-319.

Katsurada, K., Nakamura, Y., Yamada, H., & Nitta, T. (2003). XISL: a language for describing multimodal interaction scenarios. *Proceedings of the 5th international conference on Multimodal interfaces, ACM*, 281-284.

Kebritchi, M., Hirumi, A., & Bai, H. (2010). The effects of modern mathematics computer games on mathematics achievement and class motivation. *Computers & education, 55*(2), 427-443.

Kelleher, C., & Pausch, R. (2005). Stencils-based tutorials: design and evaluation. *Proceedings of the SIGCHI conference on Human factors in computing systems, ACM*, 541-550.

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM, 50*(7), 58-64.

Kluger, A. N. (1996). The effects of feedback interventions on performance: a historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological bulletin, 119*(2), 254.

Kraiger, K., Ford, J. K., & Salas, E. (1993). Application of cognitive, skill-based, and affective theories of learning outcomes to new methods of training evaluation. *Journal of applied psychology, 78*(2), 311.

Lépinard, P. (2014). Du serious gaming au full flight simulator: proposition d'un cadre conceptuel commun pour la formation des formateurs en simulation. *Systèmes d'information & management, 19*(3), 39-68.

Li, W., Grossman, T., & Fitzmaurice, G. (2012). GamiCAD: a gamified tutorial system for first time autocad users. *Proceedings of the 25th annual ACM symposium on User interface software and technology, ACM*, 103-112.

Li, W., Grossman, T., & Fitzmaurice, G. (2014). CADament: a gamified multiplayer software tutorial system. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems, ACM*, 3369-3378.

Li, W., Zhang, Y., & Fitzmaurice, G. W. (2013). TutorialPlan: Automated Tutorial Generation from CAD Drawings. *IJCAI*.

Linehan, C., Kirman, B., Lawson, S., & Chan, G. (2011). Practical, appropriate, empirically-validated guidelines for designing educational games. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM*, 1979-1988.

Malone, T. W., & Lepper, M. R. (1987). Making learning fun: A taxonomy of intrinsic motivations for learning. *Aptitude, learning, and instruction, 3*(1987), 223-253.

Mehm, F. (2010). Authoring serious games. *Proceedings of the Fifth International Conference on the Foundations of Digital Games, ACM*, 271-273.

Mehm, F., Göbel, S., & Steinmetz, R. (2011). Introducing component-based templates into a game authoring tool. *5th European Conference on Games Based Learning*, 395-403.

Mehm, F., Göbel, S., Radke, S., & Steinmetz, R. (2009). Authoring environment for story-based digital educational games. *Proceedings of the 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games, 1*, 113-124.

Mehm, F., Reuter, C., Göbel, S., & Steinmetz, R. (2012). Future trends in game authoring tools. *International Conference on Entertainment Computing, Springer Berlin Heidelberg*, 536-541.

Michael, D. R., & Chen, S. L. (2005). *Serious games: Games that educate, train, and inform.* Muska & Lipman/Premier-Trade.

Moreno-Ger, P., Burgos, D., Martínez-Ortiz, I., Sierra, J. L., & Fernández-Manjón, B. (2008). Educational game design for online education. *Computers in Human Behavior, 24*(6), 2530-2540.

Mouaheb, H., Fahli, A., Moussetad, M., & Eljamali, S. (2012). The serious game: what educational benefits? *Procedia-Social and Behavioral Sciences, 46*, 5502-5508.

Muratet, M., Torguet, P., Jessel, J. P., & Viallet, F. (2009). Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology, 2009*(3).

O'Rourke, E., Andersen, E., Gulwani, S., & Popović, Z. (2015). A framework for automatically generating interactive instructional scaffolding. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ACM*, 1545-1554.

Palmiter, S., & Elkerton, J. (1991). An evaluation of animated demonstrations of learning computer-based tasks. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems, ACM*, 257-263.

Panzoli, D., Sanselone, M., Sanchez, S., Sanza, C. C., Lagarrigue, P., & Duthen, Y. (2014). Introducing a design methodology for multi-character collaboration in immersive learning games. *Proceedings of the Sixth International Conference on Virtual Worlds and Games for Serious Applications (VS-Games14), IEEExplore digital library*.

Parker, L. E., & Lepper, M. R. (1992). Effects of fantasy contexts on children's learning and motivation: making learning more fun. *Journal of personality and social psychology, 62*(4), 625.

Pongnumkul, S., Dontcheva, M., Li, W., Wang, J., Bourdev, L., Avidan, S., & Cohen, M. F. (2011). Pause-and-play: automatically linking screencast video tutorials with applications. *Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM*, 135-144.

Prince, C., & Jentsch, F. (2001). Aviation crew resource management training with low-fidelity devices. *Improving teamwork in organizations: Applications of resource management training*, 147-164.

Querrec, R. (2003). *Les systèmes multi-agents pour les environnements virtuels de formation: application à la sécurité civile.* Brest: Doctoral dissertation.

Ramesh, V., Hsu, C., Agrawala, M., & Hartmann, B. (2011). ShowMeHow: Translating User Interface Instructions Between Similar Applications. *Proc. UIST'11*, 1-8.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67.

Reymonet, A. (2008). *Modélisation de connaissances à partir de textes pour une recherche d'information sémantique.* Thèse de doctorat. Université de Toulouse, Université Toulouse III-Paul Sabatier.

Rosenberg, M. J. (2001). *E-learning: Strategies for delivering knowledge in the digital age* (Vol. 3). New York : McGraw-Hill.

Shneiderman, B. (2004). Designing for fun: how can we design user interfaces to be more fun? *interactions, 11*(5), 48-50.

Sørensen, B. H., & Meyer, B. (2007). Serious Games in language learning and teaching–a theoretical perspective. *Proceedings of the 3rd international conference of the digital games research association*, 559-566.

Susi, T., Johannesson, M., & Backlund, P. (2007). Serious games: An overview.

Tang, S., Hanneghan, M., & Carter, C. (2013). A Platform Independent Game Technology Model for Model Driven Serious Games Development. *Electronic Journal of e-Learning, 11*(1), 61-79.

Tang, S., Hanneghan, M., Hughes, T., Dennett, C., Cooper, S., & Sabri, M. A. (2008). Towards a Domain Specific Modelling Language for Serious Game Design. *6th International Game Design and Technology Workshop, Liverpool, UK*.

Thillainathan, N. (2013). A Model Driven Development Framework for Serious Games. *SSRN 2475410*.

Thomas, R., Cahill, J., & Santilli, L. (1997). Using an interactive computer game to increase skill and self-efficacy regarding safer sex negotiation: Field test results. *Health Education & Behavior, 24*(1), 71-86.

Thompson, D., Baranowski, T., Buday, R., Baranowski, J., Thompson, V., Jago, R., & Griffith, M. J. (2008). Serious video games for health: how behavioral science guided the design of a game on diabetes and obesity. *Simulation & gaming*.

Torrente, J., Del Blanco, Á., Marchiori, E. J., Moreno-Ger, P., & Fernández-Manjón, B. (2010). <e-Adventure>: Introducing educational games in the learning process. *IEEE EDUCON 2010 Conference*, 1121-1126.

Torrente, J., Moreno-Ger, P., Fernández-Manjón, B., & Sierra, J. L. (2008). Instructor-oriented authoring tools for educational videogames. *Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on, IEEE*, 516-518.

Torrente, J., Moreno-Ger, P., Fernández-Manjón, B., & Sierra, J. L. (2008, IEEE). Instructor-oriented authoring tools for educational videogames. *Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on*, 516-518.

Torrente, J., Vallejo-Pinto, J. A., Moreno-Ger, P., & Fern, B. (2011). Introducing accessibility features in an educational game authoring tool: The< e-Adventure> experience. *Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on, IEEE*, 341-343.

Tuck, R., & Olsen, D. R. (1990). Help by guided tasks: utilizing UIMS knowledge. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 71-78.

Van Eck, R. (2007). Building artificially intelligent learning games. *Games and simulations in online learning: Research and development frameworks*, 271-307.

Van Est, C., Poelman, R., & Bidarra, R. (2011). High-level Scenario Editing for Serious Games. *GRAPP*, 339-446.

White, S. A. (2004). Introduction to BPMN. *IBM Cooperation, 2*(0), 0.

Wilson, K. A., Bedwell, W. L., Lazzara, E. H., Salas, E., Burke, C. S., Estock, J. L., & Conkey, C. (2009). Relationships between game attributes and learning outcomes review and research proposals. *Simulation & gaming, 40*(2), 217-266.

Wood, L. E., & Stewart, P. W. (1987). Improvement of practical reasoning skills with a computer game. *Journal of Computer-Based Instruction*.

Wouters, P., van der Spek, E. D., & van Oostendorp, H. (2009). Current Practices in Serious Game Research: A Review from a Learning Outcomes Perspective. *Games-based learning advancements for multisensory human computer interfaces: techniques and effective practices*, 232-255.

Wrzesien, M., & Raya, M. A. (2010). Learning in serious virtual worlds: Evaluation of learning effectiveness and appeal to students in the E-Junior project. *Computers & Education, 55*(1), 178-187.

Younis, B., & Loh, C. S. (2010). Integrating serious games in higher education programs. *Proceedings of the academic colloquium 2010: Building partnership in teaching excellence*.

Yusoff, A., Crowder, R., & Gilbert, L. (2010). Validation of serious games attributes using the technology acceptance model. *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on, IEEE*, 45-51.

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

ANNEXES

Annexe 1 : Description du déroulement de la leçon du réglage anticollision

Réglage du radar anticollision sur le véhicule Peugeot 308

I – Déroulement de la leçon

1. Dans n'importe quel ordre :

a) Établir la communication avec le véhicule (à l'aide du logiciel de diagnostic)

i. Sur l'écran de sélection de la marque du véhicule, cliquer sur Peugeot
ii. Sur l'écran de sélection du modèle du véhicule, l'apprenant a le choix entre
   - Choisir le véhicule manuellement
     - Cliquer sur le bouton 308
     - Cliquer sur le bouton OK dans la pop-up qui apparait
   - Détecter le véhicule automatiquement
     - Cliquer sur le bouton de détection automatique
     - Cliquer sur OK sur l'écran suivant une fois que le véhicule a été détecté
iii. Sur l'écran qui apparait ensuite, cliquer sur OK.
iv. Cliquer sur le bouton « Recherche de Pannes »
v. Attendre la fin de l'opération, puis sélectionner le calculateur ARTIV et cliquez sur OK.
vi. Naviguer dans DiagBox : Réparation >> Pack Réparation >> Apprentissage >> Apprentissage du radar ARTIV

b) Préparer le radar anticollision (à l'aide de l'Environnement Virtuel)

i. Démonter le pare-chocs avant
ii. Positionner automatiquement la caméra devant le radar anticollision
iii. Placer la plaque adaptée sur le radar anticollision
iv. Placer les agrafes correctement pour maintenir la plaque sur le radar
v. Calibrer l'inclinomètre par rapport au sol sur lequel est placé le véhicule
vi. Placer l'inclinomètre sur la plaque
vii. Tourner la vis A du radar jusqu'à ce que l'inclinomètre indique 90°
viii. Retirer inclinomètre, agrafes, et plaque du radar
ix. Remonter le pare-chocs avant

I – Déroulement de la leçon (suite)

2. Puis :
    a) Cliquez sur OK dans DiagBox.
    b) Sur l'écran suivant cliquer encore sur OK.
    c) Prenez connaissance des conditions requises afin d'effectuer l'apprentissage sur route du radar, puis cliquer sur OK.

3. ***Phase spéciale de roulage ➔ pendant la simulation, l'apprenant doit indiquer régulièrement quand il pense être dans les bonnes situations pour effectuer l'apprentissage. Après trois bonnes réponses, passer à la phase suivante.***

4. Activer automatiquement le Comportement Véhicule « Radar désaligné de 1.5° » afin de simuler un désalignement du radar.
5. Réglage du radar (retour en garage) :
    a) Démonter la trappe à l'avant du véhicule afin d'accéder au radar ARTIV
    b) Tourner la vis A ET B du radar du nombre de tours indiqués par le logiciel DiagBox
    c) Replacer la trappe

6. ***Deuxième phase de roulage (identique à la première).***

7. Activer automatiquement le Comportement Véhicule « Radar presque aligné ».
8. Réglage du radar (retour en garage) :
    a) Démonter la trappe à l'avant du véhicule afin d'accéder au radar ARTIV
    b) Faites 1 tour de vis antihoraire pour les vis A ET B du radar
    c) Replacer la trappe
9. Dans DiagBox, cliquer sur OK.

⇨ **Fin de la procédure**

II – Pédagogie

- Objectifs pédagogiques
    - Les étapes 1.*a) Établir la communication avec le véhicule*, et 2 sont liées à l'objectif pédagogique Savoir manipuler le logiciel de diagnostic.
    - Les étapes 1.*b) Préparer le radar anticollision*, 5 et 8 sont liées à l'objectif pédagogique Savoir opérer sur le véhicule.

- Retours pédagogiques
    - Fournir une aide associée à chaque action devant être réalisée par l'apprenant.
    - Pour chaque erreur effectuée par l'apprenant, afficher le message d'erreur générique suivant : *"Erreur : cette action ne permet pas de progresser dans l'activité en cours."*

III – Éléments ludiques

- Mission : Effectuer la procédure de réglage du radar anticollision de la Peugeot 308 d'un client qui vient d'avoir un accrochage avec un autre véhicule.

- Sous-missions : Les étapes 1.a) Établir la communication avec le véhicule, 1.b) Préparer le radar anticollision, 3, 5, 6 et 8 correspondent chacune à une sous-mission.

- Difficulté de la mission : 1 / 3

- Score requis pour valider la leçon : 80% (4 étoiles sur cinq).

- Mises en scène
  - Au début de la leçon : dialogue entre le garagiste et le client
  - A la fin de la leçon : dialogue entre le garagiste et le client

Annexe 2 : Description des éléments pédagogiques et ludiques utilisés par l'exemple introduit dans la partie 3.3.2

TABLEAU 3. DESCRIPTION DES ELEMENTS PEDAGOGIQUES

```
<PedagogicalObjective id = "objective_1">
    <Label language = "fr">
      Savoir manipuler le logiciel de diagnostic
    </ Label>
</PedagogicalObjective>

<PedagogicalObjective id = "objective_2">
    <Label language = "fr">
      Savoir opérer sur le véhicule
    </ Label>
</PedagogicalObjective>

<PedagogicalFeedback type = "help" id = "help_1">
    <Property id = "text">
        <TranslatableValue language = "fr">
          Cliquez sur le bouton Peugeot dans DiagBox,
          OU effectuez l'interaction TAKE_OFF du pare-chocs.
        </ TranslatableValue>
    </ Property>
</ PedagogicalFeedback >

<PedagogicalFeedback type = "help" id = "help_2">
    <Property id = "text">
        <TranslatableValue language = "fr">
          Cliquez sur le bouton 308 ou Détection Automatique.
        </ TranslatableValue>
    </ Property>
</ PedagogicalFeedback >

<PedagogicalFeedback type = "help" id = "help_3">
    <Property id = "text">
        <TranslatableValue language = "fr">
          Cliquez sur le bouton OK.
        </ TranslatableValue>
    </ Property>
</ PedagogicalFeedback >
```

```
<PedagogicalFeedback type = “help” id = “help_4”>
    <Property id = “text”>
        <TranslatableValue language = "fr">
          Cliquez sur le bouton OK.
        </ TranslatableValue>
    </ Property>
</ PedagogicalFeedback >

<PedagogicalFeedback type = “help” id = “help_5”>
    <Property id = “text”>
        <TranslatableValue language = "fr">
          Effectuez l’interaction PUT_PLATE_ON du radar.
        </ TranslatableValue>
    </ Property>
</ PedagogicalFeedback >

<PedagogicalFeedback type = “error” id = “global_error”>
    <Property id = “text”>
        <TranslatableValue language = "fr">
          Erreur : cette action ne permet pas de progresser
          dans l’activité en cours.
        </ TranslatableValue>
    </ Property>
</ PedagogicalFeedback >
```

TABLEAU 4. DESCRIPTION DES ELEMENTS LUDIQUES

```
<Mission difficulty = “1” validatingScore = “80” >
    <Label language = "fr">
      Effectuer le début de la procédure de réglage du radar
      anticollision de la Peugeot 308 d’un client qui vient d’avoir
      un accrochage avec un autre véhicule.
    </ Label>
</Mission>

<Sub_Mission id = “sub_mission_1”>
    <Label language = "fr">
      Établir la communication avec le véhicule.
    </ Label>
</Sub_Mission >

<Sub_Mission id = “sub_mission_2”>
    <Label language = "fr">
      Préparer le radar anticollision.
    </ Label>
</Sub_Mission >
```

```
<Staging type = “dialog” id = “dialog_1”>

    <Property id = “text_nb” value = “1”>
        <Property id = “speaker” value = “customer”> </Property>
        <TranslatableValue language = "fr">
          Bonjour Monsieur, suite à un accrochage avec un autre
          véhicule, le radar anticollision de mon véhicule ne
          fonctionne plus. Pouvez-vous m’aider ?
        </TranslatableValue>
    </Property>

    <Property id = “text_nb” value = “2”>
        <Property id = “speaker” value = “mechanic”> </Property>
        <TranslatableValue language = "fr">
          Bien sûr, je reviens vers vous dès que j’ai du nouveau.
        </TranslatableValue>
    </Property>

</Staging>

<Staging type = “dialog” id = “dialog_2”>

    <Property id = “text_nb” value = “1”>
        <Property id = “speaker” value = “mechanic”> </Property>
        <TranslatableValue language = "fr">
          La panne provenait d’un désalignement du radar.
          Tout fonctionne de nouveau !
        </TranslatableValue>
    </Property>

    <Property id = “text_nb” value = “2”>
        <Property id = “speaker” value = “customer”> </Property>
        <TranslatableValue language = "fr">
          Merci beaucoup, bonne journée !
        </TranslatableValue>
    </Property>

</Staging>
```

Annexe 3 : Liste des éléments utilisés pour décrire graphiquement les différentes dimensions abordées dans le Chapitre 4

TABLEAU 5. ELEMENTS GRAPHIQUES UTILISES POUR DECRIRE L'ACTIVITE

Bloc	Description
Début Activité	Ce bloc marque le début du déroulement des activités. Seulement un seul bloc de ce type peut être utilisé dans chaque scénario.
Fin Activité	Ce bloc marque la fin du déroulement des activités.
Attendre que le joueur appuie sur le bouton [**button_id**]	Ce bloc permet de spécifier qu'à un moment du déroulement des activités, le joueur doit appuyer sur un bouton dans le logiciel de diagnostic. Il possède une propriété, **button_id**, permettant de spécifier le bouton en question.
Attendre que le joueur effectue l'interaction [**interaction_id**]	Ce bloc permet de spécifier qu'à un moment du déroulement des activités, le joueur doit effectuer une interaction dans l'Environnement Virtuel. Il possède une propriété, **interaction_id**, permettant de spécifier l'interaction en question.
	Ce bloc permet de créer des embranchements pour ainsi définir des activités alternatives ou parallèles.
OR	Ce bloc permet de terminer un embranchement et de faire des branches associées des activités alternatives.
AND	Ce permet de terminer un embranchement et de faire des branches associées des activités parallèles.

TABLEAU 6. ELEMENTS GRAPHIQUES UTILISES POUR DECRIRE LES RETOURS PEDAGOGIQUES

Retour pédagogique	Description
	Permet de spécifier le message d'erreur à afficher aux prochaines mauvaises actions du joueur. La propriété associée est l'identifiant du message d'erreur en question.
	Permet de spécifier le message d'aide à afficher les prochaines fois que le joueur demandera une aide. La propriété associée est l'identifiant du message d'aide en question.

TABLEAU 7. ELEMENT GRAPHIQUE UTILISES POUR DECRIRE LA DIMENSION LUDIQUE

Élément ludique	Description
	Permet de spécifier qu'à un moment du déroulement des activités un dialogue se déclenchera. La propriété associée est l'identifiant du dialogue en question.

Annexe 4 : Exemple des Fichiers XML permettant de personnaliser l'Environnement Virtuel

TABLEAU 8. FICHIER XML DECRIVANT LA LISTE DES ELEMENT PRESENTS DANS L'ENVIRONNEMENT VIRTUEL

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Environments>
    <Environment id = "garage" description = "" default = "true">
        <Setting id = "garage_setting" description = ""/>
        <Element id = "bumper" description = "">
            <State id = "isOn" type = "boolean" initValue = "true"/>
            <Interaction id = "put_on">
                <StateChange environmentId = "garage" elementId =
"bumper" stateId = "isOn" newValue = "true"/>
            </Interaction>
            <Interaction id = "put_off">
                <StateChange environmentId = "garage" elementId =
"bumper" stateId = "isOn" newValue = "false"/>
            </Interaction>
        </Element>
        <Element id = "artiv" description = "">
            <Interaction id = "put_plate_on">
                <StateChange environmentId = "garage" elementId = "plate"
stateId = "isOn" newValue = "true"/>
            </Interaction>
        </Element>
        <Element id = "screw_a" description = "">
            <Interaction id = "put_screwdriver">
            </Interaction>
        </Element>
        <Element id = "screw_b" description = "">
            <Interaction id = "put_screwdriver">
            </Interaction>
        </Element>
        <Element id = "plate" description = "">
            <State id = "isOn" type = "boolean" initValue = "false"/>
            <Interaction id = "put_plate_off">
                <StateChange environmentId = "garage" elementId = "plate"
stateId = "isOn" newValue = "false"/>
            </Interaction>
            <Interaction id = "put_staples_on">
            </Interaction>
        </Element>
        <PoV id = "front" description = "" default = "true">
            <SettingInstance textureId = "garage.front.setting"/>
            <ElementInstance associatedElementId = "bumper">
                <Sprite id = "default" textureId =
"garage.front.bumper.on_car" orderInLayer = "1" default = "true" />
            </ElementInstance>
        </PoV>
        <PoV id = "front_2" description = "" default = "false">
            <SettingInstance textureId = "garage.front_2.setting"/>
            <ElementInstance associatedElementId = "artiv">
                <Sprite id = "default" textureId = "garage.front_2.artiv"
orderInLayer = "1" default = "true" />
```

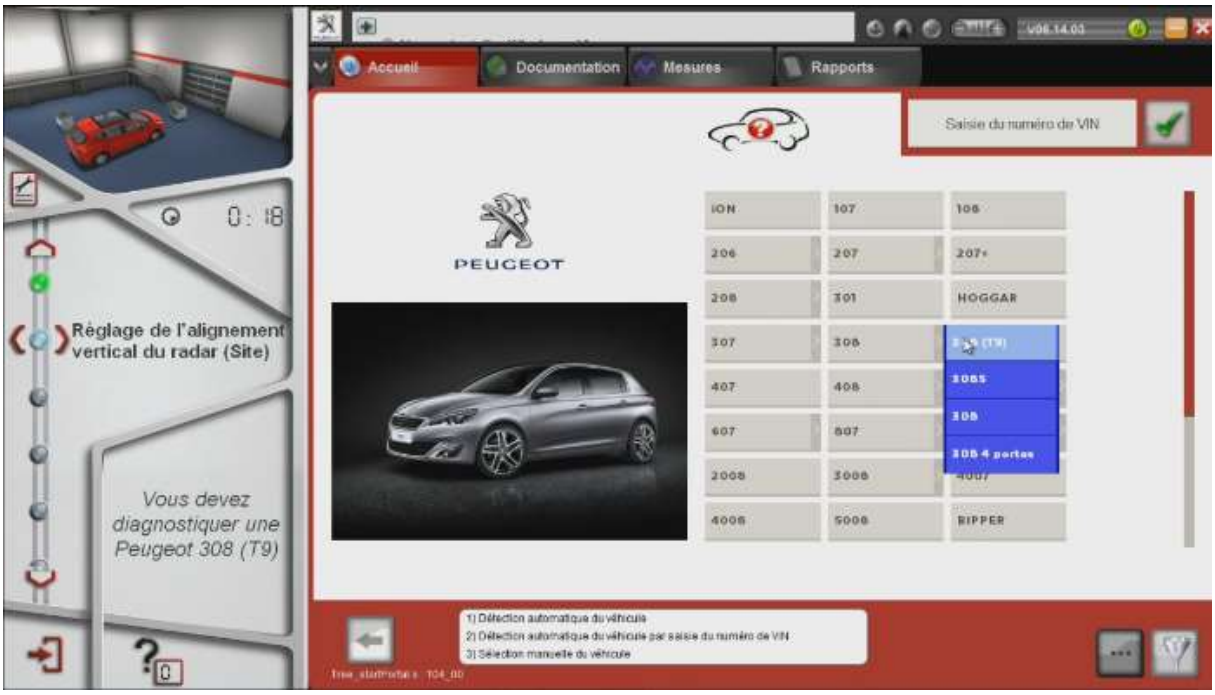
```
            </ElementInstance>
            <ElementInstance associatedElementId = "screw_a">
                <Sprite id = "default" textureId =
"garage.front_2.artiv.screw_a" orderInLayer = "2" default = "true" />
            </ElementInstance>
            <ElementInstance associatedElementId = "screw_b">
                <Sprite id = "default" textureId =
"garage.front_2.artiv.screw_b" orderInLayer = "2" default = "true" />
            </ElementInstance>
            <ElementInstance associatedElementId = "plate">
                <Sprite id = "default" textureId = "garage.front_2.plate"
orderInLayer = "4" default = "true" />
            </ElementInstance>
            <ElementInstance associatedElementId = "bumper">
                <Sprite id = "default" textureId =
"garage.front_2.bumper.on_car" orderInLayer = "5" default = "true" />
            </ElementInstance>
        </PoV>
    </Environment>
</Environments>
```

TABLEAU 9. FICHIER XML DECRIVANT LA LISTE DES TEXTURES UTILISEES POUR AFFICHER LES DIFFERENTS ELEMENTS DE L'ENVIRONNEMENT VIRTUEL

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Textures>
    <Texture id="garage.front.setting"
path="extern/textures/garage/front/setting.png"/>
    <Texture id="garage.front.bumper.on_car"
path="extern/textures/garage/front/bumper/on_car.png"/>
    <Texture id="garage.front_2.setting"
path="extern/textures/garage/front_2/setting.png"/>
    <Texture id="garage.front_2.bumper.on_car"
path="extern/textures/garage/front_2/bumper/on_car.png"/>
    <Texture id="garage.front_2.artiv"
path="extern/textures/garage/front_2/artiv/artiv.png"/>
    <Texture id="garage.front_2.artiv.screw_a"
path="extern/textures/garage/front_2/artiv/screw_a/screw_a.png"/>
    <Texture id="garage.front_2.artiv.screw_b"
path="extern/textures/garage/front_2/artiv/screw_b/screw_b.png"/>
    <Texture id="garage.front_2.plate"
path="extern/textures/garage/front_2/plate/plate.png"/>
</Textures>
```

Annexe 5 : Captures d'écran du Projet Diag'Adventures

Les captures d'écrans présentées ci-dessous ont été prises et fournies par la société Opérantis[45].



[45] http://www.operantis.fr/societe/qui-sommes-nous/

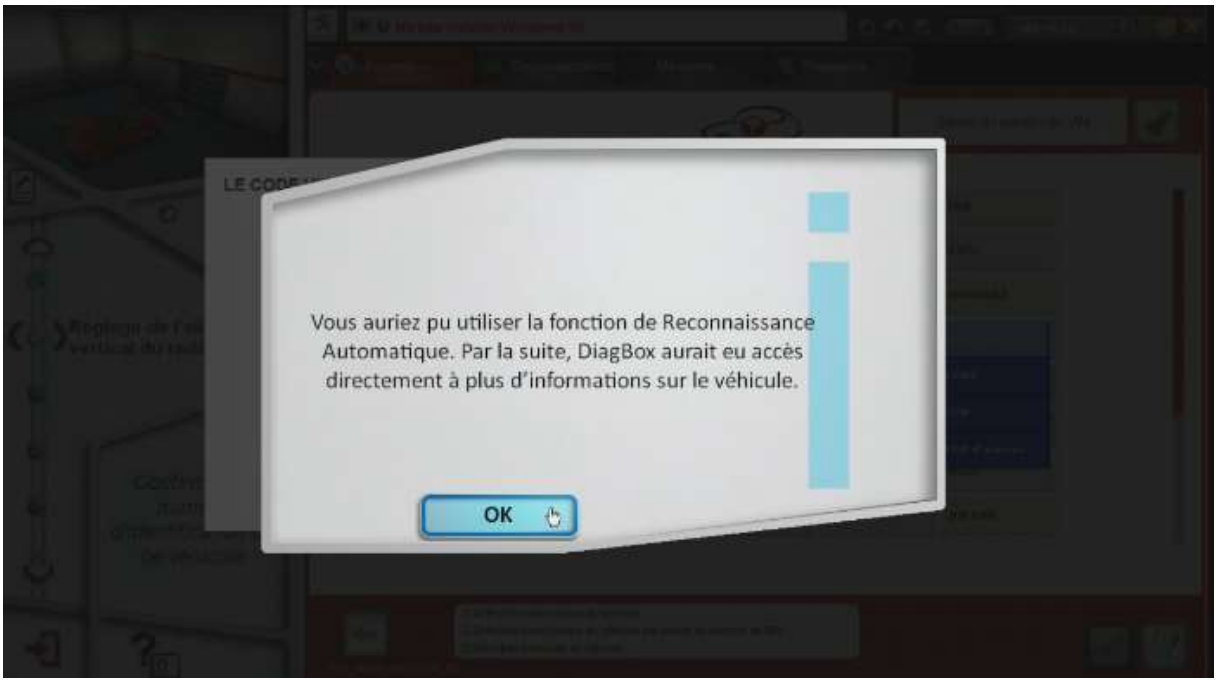


FIGURE 48. PROJET DIAG'ADVENTURES : PLUSIEURS CAPTURES D'ECRAN PRESENTANT LE RESULTAT OBTENU AU TERME DES TRAVAUX

Annexe 6 : Questionnaire fourni aux garagistes afin d'évaluer notre outil

ACTIA®

Enquête de satisfaction

Cher client,

Dans le cadre de la mise en place de notre outil de formation, nous visons une amélioration continue de nos services.

Nous vous demandons quelques minutes de votre précieux temps afin de remplir ce questionnaire et nous le transmettre à :

ACTIA Automotive, 5 rue Jorge Semprun, BP74215 – Toulouse 31432 Cedex 4.

Votre avis nous intéresse et nous permettra de déterminer l'efficacité du jeu et d'améliorer

Vous …

Questionnaire rempli par : _____ *Date :* Cliquez ici pour entrer une date.

Nom de l'entreprise : _____ *Code RRDI :* _____

Adresse : _____ *N° de téléphone :* _____

Code Postal : _____ *Ville :* _____

Vous êtes ☐ *Employé polyvalent* ☐ *Mécanicien* ☐ *Carrossier* ☐ *Référent technique* ☐ *Expert technique* ☐ *Autres :* _____

Aviez-vous déjà joué à un Serious Game ? ☐ *Oui* ☐ *Non* ☐ *Ne se prononce pas*

Votre expérience de jeu …	😃😃	😃	😐	☹️	☹️☹️
Le « Jeu Sérieux », était-il plaisant à jouer ?	☐	☐	☐	☐	☐
Avez-vous aimé les graphismes de jeu ?	☐	☐	☐	☐	☐
Comment avez-vous trouvé l'intuitivité du jeu ?	☐	☐	☐	☐	☐
Comment avez-vous trouvé la fluidité du jeu ?	☐	☐	☐	☐	☐
Comment avez-vous trouvé la durée du jeu ?	☐	☐	☐	☐	☐
Les explications vous ont elles semblées claires ?	☐	☐	☐	☐	☐

Vos notes … (1 : Très mauvaise / 10 : Très bien)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Note globale	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
L'immersion dans l'univers du jeu	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
La difficulté du jeu	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Votre compréhension des enseignements du jeu	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Votre score …	**Vos Commentaires …**
Nombre d'aides :	Avez-vous constaté des manques dans ce Serious Game ?
Durée de la partie	Quelles améliorations, proposeriez-vous afin d'avoir une meilleure expérience de jeu ?

Seriez-vous intéressé par des formations guidées au travers du Jeu Sérieux ?

☐ **Oui** ☐ **Non** ☐ **Ne se prononce pas**

Table des Figures

Table des Tableaux