# Système tutoriel intelligent pour l'apprentissage de travail procédural et collaboratif

Cédric Buche buche@enib.fr
Pierre De Loor deloor@enib.fr
Ronan Querrec querrec@enib.fr

Laboratoire d'Ingénierie Informatique
Centre Européen de Réalité Virtuelle
BP 38 - 29280 Plouzané – FRANCE

**Résumé :**
Cet article s'inscrit dans la formalisation des processus à mettre en œuvre pour le développement d'environnements virtuels de formation différentiée. Il se situe plus précisément dans le cadre de l'apprentissage de prise de décision au sein de procédures collaboratives, et s'attache à définir l'implémentation, sous forme d'agents, de modèles fondamentaux des systèmes tutoriels intelligents. Il précise les caractéristiques comportementales et fonctionnelles liées à la problématique procédurale et à l'utilisation d'environnements virtuels immersifs.

**Mots-clés :** Environnement virtuel de formation, pédagogie, agent.

**Abstract:**
This article takes place in the formalization of processes for the development of differentiated virtual environments for training. It defines the implementation, in the form of agents, of basic models of the intelligent tutoring systems, considering the learning of decisional behaviour within collaboratives procedures. It specifies the behavioral and functional features linked to the procedural problematic and to the usage of immersive virtual environments.

**Keywords:** Virtual environment for training, education, agent.

## 1 Introduction

Nos travaux s'articulent autour de l'utilisation de la réalité virtuelle pour la mise en œuvre d'une pédagogie différenciée axée sur la formation à la prise de décision au sein d'un travail procédural collaboratif. Un formateur encadre plusieurs apprenants. Dans ce contexte, chaque étudiant est immergé dans un environnement virtuel qui simule son environnement physique et rend compte de son environnement social [1]. Les procédures agencent des actions exécutables dans l'environnement virtuel.

Nous distinguons le rôle du formateur, spécialiste du domaine, de celui du pédagogue, expert en didactique. Lors de l'utilisation de notre système, c'est le formateur qui est en relation avec des apprenants et sa charge augmente avec le nombre d'apprenants. Notre objectif est alors de simuler un raisonnement pédagogique didacticien, afin de proposer au formateur des assistances pédagogiques, adaptées au contexte de simulation et à l'apprenant. Pour cela, le système analyse les interactions entre l'apprenant et l'environnement virtuel et interagit avec le formateur pour choisir l'assistance pédagogique la plus pertinente (figure 1).

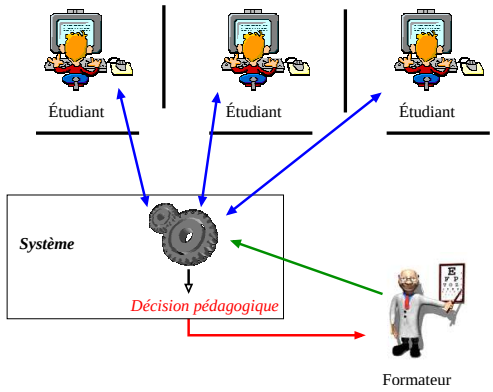


FIG. 1 – Le système interagit avec des apprenants et un formateur.

Cet article présente un système intégrant les connaissances sur l'apprenant, le domaine, l'interface (interaction système / apprenant) et la pédagogie qui sont réifiées et mises à jour en cours de simulation. Pour cela, nous proposons un système tutoriel intelligent (Intelligent Tutoring System), où les composantes sont représentées sous la forme d'un système multi-agents. Les connaissances représentées servent de base au raisonnement effectué par un agent pédagogique (ne figurant pas dans cet article).

Même s'il existe quelques applications qui intègrent un ITS dans un environnement virtuel de formation, la plupart ne disposent pas de connaissances explicites utilisables pédagogiquement et se contentent de simuler l'environnement [4]. D'autres n'incorporent qu'une représentation des connaissances sur le domaine et/ou l'apprenant [6]. Les plus évolués proposent un modèle de diagnostic [5], où la notion d'erreur et les assistances proposées sont spécifiés pour chaque exercice (et donc non générique).

## 2 Les modèles

Dans cette section, nous montrons comment nous représentons les connaissances qui servent de base au raisonnement pédagogique de notre système. Ces connaissances sont contenues dans des modèles inspirés de ceux qui sont utilisés classiquement dans un ITS [1] :

1. le *modèle du domaine*, représentant la connaissance, utile pédagogiquement, de l'expert sur le domaine ;
2. le *modèle de l'apprenant*, permettant d'établir l'état de ses connaissances à un instant donné ainsi q'un profil ;
3. le *modèle des erreurs*, proposant d'identifier les erreurs et contenant une base de connaissances sur les erreurs classiques ;
4. le *modèle d'interface*, permettant l'échange d'informations entre le système et l'utilisateur ;
5. le *modèle pédagogique*, prenant les décisions d'enseignement.

Chacun de ces modèles se base sur un modèle (`AgentPackage`) plus abstrait d'agent (`Agent`), de comportement (`Behavior`) et de connaissances (`KB`). Les agents sont en interaction, ce qui lui permet de mettre à jour ces connaissances et d'apater son comportement. Ils fournissent les connaissances qui sont utilisées pour effectuer un raisonnement pédagogique, rôle endossé par le modèle pédagogique qui n'est pas traité dans cet article.

Le formateur intervient en utilisant le `TeacherAgent`. Cet agent n'est pas associé à un modèle.

### 2.1 L'interaction inter-modèles

Les modèles interagissent et s'échangent des données qui peuvent être extraites de la simulation ou déduites d'un raisonnement interne au modèle. Nous proposons un processus en 5 étapes, mis en place pour chaque apprenant (figure 2) :

1. *Observer* (`InterfaceAgent`) : utilisant le modèle de l'interface, le système analyse les activités de l'apprenant. Les éléments pertinents pour la formation sont fournis au modèle de l'apprenant. Ces informations concernent :
   – les actions de l'apprenant (`PCVAction` voir section 2.5)
   – les éléments observables par l'apprenant (`PCVObservation` voir section 2.5)
   – les déplacements de l'apprenant (`PCVMotion` voir section 2.5)
2. *Détecter et Identifier une erreur* (`LearnerAgent`, `ExpertAgent`, `ErrorAgent`) : le système analyse les actions de l'apprenant (modèle de l'apprenant) et les compare aux actions à effectuer (modèle du domaine). Cette confrontation permet de détecter une éventuelle erreur. Si une erreur a été détectée, un mécanisme d'identification de l'erreur est mis en place (en utilisant le modèle des erreurs). De plus, les règles métiers du modèle du domaine sont vérifiées (règles indépendantes de l'ordonnancement de la procédure).
3. *Proposer des assistances pédagogiques* (`PedagogicalAgent`) : utilisant le modèle de l'apprenant (caractéristiques, activités, erreur) et le modèle du domaine (connaissances sur les structures organisationnelles), un mécanisme simulant un raisonnement pédagogique préconise les assistances pédagogiques adaptées à la situation.
4. *Choisir une assistance pédagogique* (`TeacherAgent`) : le formateur sélectionne une assistance pédagogique parmis celles qui ont été préconisées.
5. *Représenter l'assistance pédagogique* (`InterfaceAgent`) : l'assistance pédagogique sélectionnée est représentée dans l'environnement virtuel.

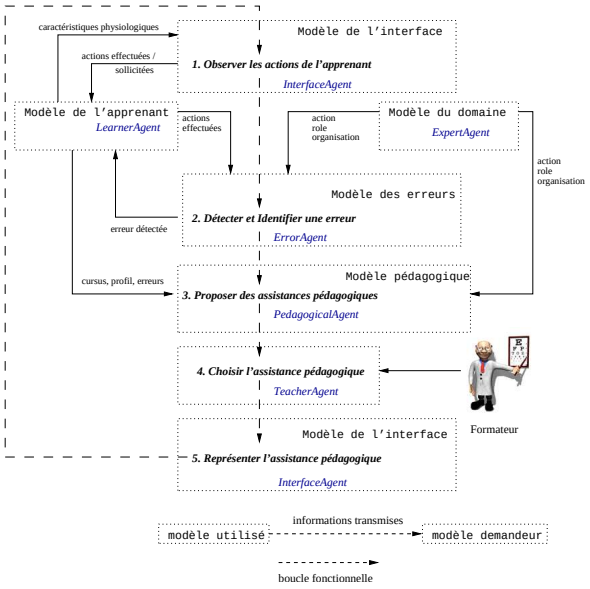


FIG. 2 – Le processus pédagogique de notre système est constitué de 5 étapes.

## 2.2 Le modèle du domaine

Le modèle du domaine est une représentation de la connaissance que l'apprenant doit acquérir, il représente l'ensemble des connaissances expertes sur le domaine d'apprentissage. Ce modèle offre au reste du système la possibilité, d'une part de vérifier si les actions effectuées par l'apprenant sont correctes, et d'autre part de proposer des solutions ou des explications aux problèmes rencontrés par l'apprenant. Il contient également des informations permettant d'interpréter ces connaissances, afin de permettre l'exécution des actions pour « *faire à la place* »de l'apprenant. Le modèle du domaine contient alors des connaissances qui sont comparables aux connaissances déclaratives et procédurales misent en jeu lors de l'apprentissage de compétences en psychologie.

**Base de connaissances** Dans le cas de l'apprentissage de procédures collaboratives en milieu réel, ces connaissances portent sur l'environnement social et sur les procédures. Le modèle du domaine que nous proposons s'appuie sur le modèle d'organisation de `MASCARET` [1]. Nous distinguons les connaissances sur :

1. *L'environnement social*
   La connaissance sur l'environnement social est constituée d'un ensemble de connaissances expertes sur les équipes. Chaque connaissance experte utilise la connaissance sur la structure organisationnelle du modèle (`Team` dans `MASCARET`), ce qui lui permet d'avoir une représentation des rôles, des relations hiérarchiques entre rôles, et des responsabilités au sein des procédures.
2. *Les procédures*
   La connaissance sur les procédures permet la définition de l'ordonnancement des actions.
3. *Les règles d'usage général*
   Le domaine est également constitué de règles indépendantes des procédures (`Rule`) (figure 3). Ces règles sont à respecter et ne sont pas soumises à l'ordonnancement de la procédure.

**Représentation et interprétations des connaissances**

1. *Les équipes et les procédures*
   La représentation des connaissances sur les équipes et les procédures utilisent la structure organisationnelle définit dans `MASCARET`. Le modèle du domaine maintient des connaissances générales indépendantes du contexte dans lequel elles sont utilisées. Ainsi le modèle que nous proposons, s'appuie sur les organisations de `MASCARET`, mais ne référence pas les instances de ces organisations. Il utilise uniquement la structure organisationnelle qui représente ses instances. Ainsi, un agent sollicitant l'agent maintenant le modèle du domaine (`ExpertAgent`), ne demande pas des informations spécifiques (par exemple : Faut il faire l'action « marcher vers l'objet $O$ » ?), mais pose une question générique (par exemple « quelle action $A$ doit être effectuée, après l'action $B$, dans la mission $M_1$, considérant le rôle $R_1$ et dans l'organisation $O_1$ ? »). La réification du concept d'organisation dans `MASCARET` permet d'effectuer un tel raisonnement.
2. *Les actions et les règles d'usage général*
   La représentation des pré/post conditions d'une action et la représentation des règles d'usage sont sous forme de règles logiques.

Nous souhaitons obtenir des explications sur les procédures, actions ou règles d'usage (telle condition est non remplie). Les connaissances représentées doivent avoir une capacité d'interprétation et de description proche du langage naturel (afin de pouvoir être décrite par un spécialiste non informaticien). Par conséquent, la représentation des règles dans la base de connaissances utilise un formalisme symbolique et l'interprétation utilise une méthode de raisonnement basé sur un chaînage avant ou arrière, selon les cas d'explications attendues. Les règles constituent une base de connaissances et sont réifiées afin d'être utilisées par un moteur d'explications. Nous proposons qu'une règle soit constituée de composantes correspondant à des termes `Prolog` (`LogicalTerm`). La règle peut être évaluée, et l'évaluation des termes permet de générer une explication le cas échéant.

**Comportements** L'`ExpertAgent` propose des comportements réactifs (Diag. de classe UML figure 3) liés aux éléments suivants :

1. l'*environnement social* (`OrganisationBehavior`).
   L'`ExpertAgent` possède la capacité de raisonnement sur les agents et les rôles. Il est capable de renseigner sur les responsabilités de chaque intervenant. En considérant la connaissance sur l'environnement social, il répond à la question « est ce que l'action A est de la responsabilité du rôle R ? ».
2. les *procédures* (`ProcedureBehavior`).
   Le modèle `MASCARET` propose un modèle de

procédure (agencement d'actions). Comme STEVE [6], l'ExpertAgent est capable de fournir des explications. Il peut fournir l'ordonnancement des actions ainsi que les buts de chaque étape (procédure / action).

3. les *actions* (ActionBehavior).
   Le modèle MASCARET propose un modèle d'actions soumis à des pré/post conditions. Ces dernières sont représentées sous la forme de règles. Comme STEVE, l'ExpertAgent est capable de fournir des explications et de faire à la place de l'apprenant. Il peut fournir les pré/post conditions et expliquer pourquoi elles sont (ou ne sont pas) satisfaites.
4. les *règles du domaine* (RuleBehavior) .
   L'ExpertAgent a la possibilité de fournir des explications sur les « règles d'usage »(indépendantes de la procédure) (Rule) en s'appuyant sur les réifications des termes de la règle.

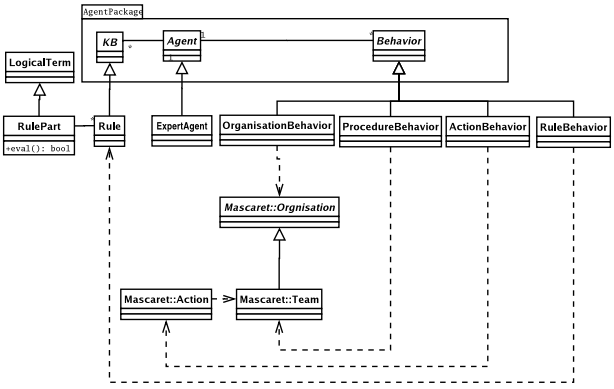


FIG. 3 – Modèle de l'expert.

## 2.3 Le modèle des erreurs

L'objectif du modèle des erreurs est de représenter la connaissance permettant d'expliquer pourquoi il y a erreur et d'associer des éléments pouvant être utilisés pour faciliter leur compréhension.

**Base de connaissances** Les connaissances portent sur une caractérisation générique des erreurs permettant de les détecter et de fournir à l'agent pédagogique des moyens d'y réagir (explication, alerte ...). La base de connaissances est constituée d'une liste d'erreurs classiques du domaine.

**Représentation et interprétations des connaissances** L'objectif, exposé précédemment, nous a amené à typer les erreurs. Les différents types distinguent les erreurs sur les règles d'usage (UsageError), sur les rôles (TeamError : rôle dans l'équipe), sur la procédure (ProceduralError : ordonnancement) et sur les actions (ActionError : pré-conditions).

Des informations supplémentaires, considérant les erreurs classiques du domaine, peuvent être écrites par le formateur et prendre la forme d'une règle. La connaissance est alors représentée en spécifiant les caractéristiques de ces erreurs classiques (partie gauche de la règle) et des informations supplémentaires associés (partie droite de la règle).

Un exemple de règle est « si (l'étudiant effectue l'action C après l'action A) ; (c'est une erreur sur la procédure parce que Z) ». « Z »étant une règle constituée d'un texte et des entités de l'environnement virtuel (Object3D).

**Comportements** L'ErrorAgent propose les comportements suivants (figure 4) :

1. *Détecter et typer les erreurs (ErrorDetectionBehavior)* :
   Il détecte une erreur potentielle effectuée par l'étudiant :
   – il vérifie que l'action sollicitée par l'apprenant est faisable (pré-conditions)(ActionError) .
   – Il compare les actions possibles pour avancer d'un pas dans la procédure (fournies par le modèle du domaine) et l'action sollicitée par l'apprenant (fournie par le modèle de l'apprenant).
   Les informations sur ces actions sont récupérées en dialogant respectivement avec l'ExpertAgent et le LearnerAgent associé à l'apprenant. Il s'agit de vérifier si l'action sollicitée fait partie des actions possibles (ProceduralError). Il s'assure également que l'agent en charge de l'action sollicitée a bien la possibilité (définit par son rôle) d'effectuer l'action sollicitée (TeamError).
   – Il vérifie les règles métiers du modèle du domaine en utilisant une base de faits issue de l'environnement virtuel fournie par l'InterfaceAgent(UsageError) .

   Lorsque l'ErrorAgent a détecté et typé une erreur, il la transmet à l'agent pédagogique.
2. *Reconnaître une erreur classique du domaine (ErrorAssociationBehavior)* :
   Notre modèle des erreurs contient une base

de connaissances sur les erreurs classiques effectuées par un étudiant dans le domaine concerné.

L'`ErrorAgent` est capable de reconnaître une telle erreur en comparant l'erreur a la partie gauche des règles correspondant aux erreurs connues de la base de connaissances. Il enrichi alors l'information sur l'erreur (qui a été préalablement typé) avec des éléments supplémentaires (`Concept_on_Error`).

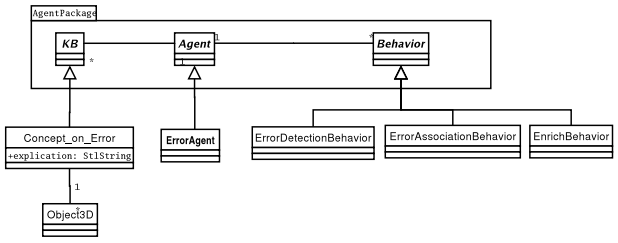


FIG. 4 – Modèle des erreurs.

3. *S'auto enrichir (`EnrichBehavior`)*
   L'`ErrorAgent` enregistre des informations sur la fréquence des erreurs afin d'en rendre compte au formateur. Un mécanisme, sous le contrôle du formateur, utilise cette information pour enrichir la connaissance sur les erreurs classiques du domaine.

**Communication** Lorsque l'`InterfaceAgent` détecte une nouvelle action de l'apprenant, il informe l'`ErrorAgent`. Ce dernier analyse la dernière action de l'apprenant, en sollicitant le `LearnerAgent` et en consultant l'`ExpertAgent` pour connaître l'action à effectuer. L'`ErrorAgent` instancie une `Error` qui est communiquée au `LearnerAgent`.

## 2.4 Le modèle de l'apprenant

Le modèle de l'apprenant doit représenter les informations caractérisant l'étudiant tant d'un point de vue prototypique (profil de l'étudiant) que d'un point de vue cursus (progression de l'étudiant). L'objectif n'est pas de modéliser un humain virtuel, mais plutôt de récolter des informations dans un cadre de formation.

**Base de connaissances** Nous avons utilisé la distinction proposée par [2], séparant dans la base de connaissance la partie *épistémique* et la partie *non épistémique* :

1. La partie *épistémique* (`EpistemicKB`) fournit les informations sur l'état des connaissances de l'apprenant pour les notions présentes dans le domaine. Elle contient une représentation des étapes et des erreurs effectuées par l'apprenant. Ces informations sont mises à jour dynamiquement et sont rendues disponibles par le `LearnerAgent`.
   Comme dans la proposition de Lourdeaux [5], notre modèle de l'apprenant contient un sous-modèle du domaine en représentant les étapes effectuées. Nous faisons alors l'hypothèse simplificatrice considérant que les actions reflètent les acquis de l'apprenant. De plus, il nous semble pertinent d'y ajouter des informations sur les erreurs effectuées. Les actions et les erreurs de l'apprenant sont stockées en suivant le déroulement de la simulation, nous obtenons alors son cursus lors de l'exercice.
2. La partie *non épistémique* (`NonEpistemicKB`) propose une représentation sur les informations tirées de l'ingénierie cognitive : *psychologiques*, *physiologiques* (caractéristiques auditives et visuelles de l'étudiant utilisées par le `PCVObservation` voir 2.5) et une *mémoire* des éléments potentiellement observés par l'apprenant (de manière instantanée (`sensorialTimeMemory`) ou à court terme sur les dernières secondes (`shortTimeMemory`))

**Mise à jour des connaissances** Pour chaque apprenant, un `InterfaceAgent` et un `LearnerAgent` sont utilisés. L'`InterfaceAgent` met à jour le modèle de l'apprenant en lui indiquant les activités de l'apprenant.

**Comportements**

1. Le `LearnerAgent` met à jour les informations dynamiques concernant les activités de l'apprenant ( figure 5 )
   (a) `CurriculumBehavior` : nouvelle action / erreur ;
   (b) `MemoryBehavior` : les objets ayant une représentation physiques et qui ont pu être observés par l'apprenant. Ces informations sont mises à jour en utilisant le `PCVObservation`.
2. Le `LearnerAgent` répond aux requêtes d'autres agents (`InformBehavior`) concernant les activités (exemple : dernière action effectuée) ou particularités de l'apprenant

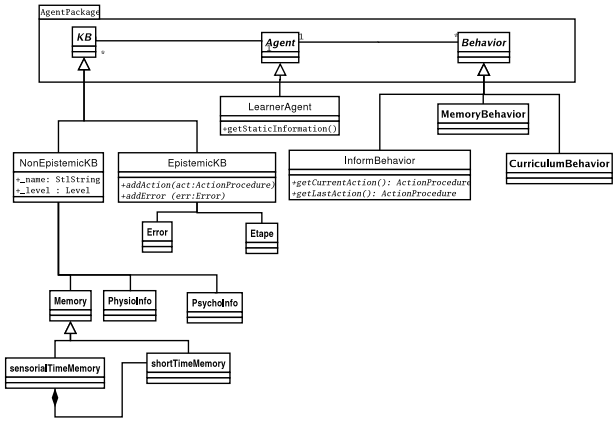


FIG. 5 – Modèle de l'apprenant.

(exemple : stratégie d'apprentissage privilégiée).

### 2.5 Le modèle de l'interface

Le modèle de l'interface a la charge de la communication bidirectionnelle entre l'apprenant et le système. Dans un cadre de formation, il s'agit d'offrir à l'apprenant la possibilité d'interagir avec son environnement et d'analyser ses activités. Nous nous inspirons des Primitives Comportementales Virtuelles (PCV) de Fuchs [3]. L'auteur les regroupe en quatre catégories :
– observer le monde virtuel ;
– se déplacer dans le monde virtuel ;
– agir sur le monde virtuel ;
– communiquer avec autrui.

**Base de connaissances** Notre modèle de l'interface contient une base de connaissances sur l'apprenant (`LearnerInformationKB`, sous partie de la base de connaissance du modèle de l'apprenant) à observer permettant de prendre en compte ses caractéristiques personnelles. Ces informations proviennent du dialogue entre l'`InterfaceAgent` et le `LearnerAgent` (modèle de l'apprenant). Des règles d'ergonomies pourraient enrichir le modèle.

**Comportements** L'`InterfaceAgent` possède trois comportements qui correspondent à chaque PCV. Nous n'avons pas traité la PCV communication, car les connaissances sur ce point semblent pour l'instant délicat à exploiter pédagogiquement. Le `PCVAction` propose à l'apprenant d'interagir avec les objets de l'environnement et détecte les actions de l'apprenant. Les `PCVObservation` et `PCVMotion` analysent ses activités.

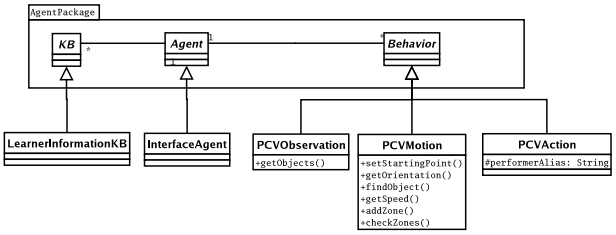


FIG. 6 – Le package PCV : observation, déplacement et action.

## 3 Bilan

Nous avons présenté les modèles de notre ITS qui contiennent les connaissances qui vont servir de base au raisonnement pédagogique de notre système. Ces connaissances sont contenues dans des modèles inspirés de ceux qui sont utilisés classiquement dans un ITS.

Nous avons détaillé les modèles du domaine, de l'apprenant, des erreurs et de l'interface. Nous avons présenté la dynamique entre les modèles, qui permet de mettre à jour les connaissances de chacun et définit un processus amenant à une décision pédagogique.

## Références

[1] C. Buche, R. Querrec, P. De Loor, and P. Chevaillier. MASCARET : A pedagogical multi-agent system for virtual environment for training. *International Journal of Distance Education Technologies*, 2(4) :41–61, Novembre 2004.

[2] N. Delestre. *METADYNE, un Hypermèdia Adaptatif Dynamique pour l'Enseignement*. PhD thesis, Université de Rouen, 2000.

[3] P. Fuchs and G. Moreau. *Le traité de la réalité virtuelle*. Presses de l'école des mines, 2003.

[4] P. Levesque. Creation and use of 3d as-built models at edf. In *FIG Working Week 2003*, 2003.

[5] D. Lourdeaux. *Réalité virtuelle et formation : conception d'environnement virtuels pédagogiques*. PhD thesis, Ecole des mines de Paris, 2001.

[6] J. Rickel and L. Johnson. Animated agents for procedural training in virtual reality : Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13, 1999.