Une assistance à l'interaction 3D en réalité virtuelle par un raisonnement sémantique et une conscience du contexte

Yannick Dennemont

Mémoire pour l'obtention du titre
de Docteur de l'Université d'Evry-Val-d'Essone
Spécialité : Informatique

Une assistance à l'interaction 3D en réalité virtuelle par un raisonnement sémantique et une conscience du contexte

Yannick Dennemont

8 Juillet 2013

Laboratoire Informatique, Biologie Intégrative et Systèmes Complexes
Équipe Interactions, Réalité Augmentée, Robotique Ambiante

Jury

P. Bourdot	Directeur de recherche, LIMSI-CNRS, Orsay	Rapporteur
I. Thouvenin	Enseignant-chercheur, HDR Heudiasyc, UTC, Compiègne	Rapporteur
J. Tisseau	Professeur, ENIB, Brest	Examinateur
M. Mallem	Professeur, UEVE, Evry	Directeur de thèse
S. Otmane	Professeur, UEVE, Evry	Encadrant
G. Bouyer	Maître de conférence, ENSIIE, Evry	Encadrant

RÉSUMÉ

Les tâches dans les environnements virtuels immersifs sont associées à des techniques et à des dispositifs d'interaction 3D (e.g. la sélection d'objets 3D à l'aide de la main virtuelle via un flystick). Alors que les environnements et les tâches deviennent de plus en plus complexes, les techniques ne peuvent plus être les mêmes pour chaque application, voire pour les différentes situations au sein d'une application. Une solution est d'adapter l'interaction en fonction des besoins de la situation pour améliorer l'utilisabilité. Ces adaptations peuvent être effectuées manuellement par le concepteur ou l'utilisateur, ou automatiquement par le système créant ainsi une interaction adaptative.

La formalisation d'une telle assistance automatique nécessite la gestion d'informations pertinentes au vu de la situation. L'ensemble de ces informations fait émerger le contexte de l'interaction. L'assistance adaptative obtenue en raisonnant à partir de ces informations est ainsi consciente du contexte. De nombreuses possibilités existent pour l'obtenir. Notre objectif est une gestion du contexte qui préserve ses degrés élevés d'expressivité et d'évolutivité tout en étant facile à intégrer.

Nous proposons une modélisation de ce problème par des graphes conceptuels basés sur une ontologie et gérés par un moteur externe en logique du premier ordre. Le moteur est générique et utilise une base de connaissance contenant des faits et des règles, qui peuvent être changés dynamiquement. Nous avons intégré une notion de confiance, afin d'établir l'adéquation d'une situation à la base de connaissances. La confiance des réactions est comparée à leur impact afin de ne garder que les pertinentes tout en évitant de saturer l'utilisateur.

Les applications utilisent des outils qui peuvent être contrôlés par le moteur. Des capteurs permettent d'extraire des informations sémantiques pour le contexte. Des effecteurs permettent d'agir sur l'application et d'obtenir des adaptations. Un jeu d'outils et une base de connaissance pour l'interaction 3D ont été créés. De nombreuses étapes sont introduites dans la base de connaissance pour de bonnes combinaisons et une réflexion indépendante d'outils spécifiques.

Nos premières applications illustrent la compréhension de la situation, dont les intérêts et difficultés de l'utilisateur, et le déclenchement d'assistances adaptées. Une étude hors ligne montre ensuite l'accès et l'évolution des étapes du moteur selon la situation. Le raisonnement sémantique générique obtenu est alors expressif, compréhensif, extensif et modifiable dynamiquement. Pour l'interaction 3D, il permet une assistance universelle automatique, ponctuelle ou manuelle à l'utilisateur et des analyses hors-lignes d'activités ou de conceptions pour le concepteur.

Mots clés : Réalité Virtuelle, Interaction 3D, Graphes conceptuels, Raisonnement sémantique, Conscience du Contexte, Adaptativité.

ABSTRACT

Tasks in immersive virtual environments are associated with 3D interaction techniques and devices (e.g. the selection of 3D objects with the virtual hand and a flystick). As environments and tasks become more and more complex, techniques can not remain the same for each application, even for every situations of a single application. A solution is to adapt the interaction depending on the situation in order to increase usability. These adaptations can be done manually by the designer or the user, or automatically by the system thus creating an adaptative interaction.

Formalisation of such assistance needs the management of pertinent information regarding the situation. Those items of information make the context emerge from the interaction. The adaptative assistance obtained by reasoning on this information is then context-aware. Numerous possibilities can be used to build one. Our objective is a context management that preserves its high degrees of expressiveness and evolutivity while being easy to plug in.

We have built a model for this issue using conceptual graphs based on an ontology and managed externally with a first order logic engine. The engine is generic and uses a knowledge base with facts and rules which can be dynamically changed. We have added a confidence notion, in order to establish a situation similarity to the knowledge base. Reactions' confidences are compared to their impacts so as to keep only the pertinent ones while avoiding user overload.

Applications have tools that can be controlled by the engine. Sensors are used to extract semantic information for the context. Effectors are used to act upon the application and to have adaptations. A tools set and a knowledge base have been created for 3D interaction. Numerous steps have been added in the knowledge base to obtain good combinations and a reasoning independent from specific tools.

Our first applications shows the situation understanding, including user interests and difficulties, and the triggering of pertinent assistances. An off-line study illustrates the access and evolution of the internal engine steps. The built generic semantic reasoning is expressive, understandable, extensive and modifiable dynamically. For 3D interaction, it allows universal assistances for the user that can be automatic, punctual or manual and off-line activities or conceptions analysis for the designers.

Keywords : Virtual Reality, 3D Interaction, Conceptual Graphs, Semantic reasoning, Context-awareness, Adaptativity.

PUBLICATIONS

Une part des idées et des schémas est apparu précédemment dans les publications ci-contre :

Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. 3D interaction assistance in virtual reality : a semantic reasoning engine for contexta-wareness : from interests and objectives detection to adaptations. In *Proc. of the 8th International Conference on Computer Graphics Theory and Applications (GRAPP 2013)*, pages 349-358, Barcelone, Espagne, 2013.

Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. A Semantic Reasoning Engine for Context-Awareness : Detection and Enhancement of 3D Interaction Interests. *In Poster Session of the Proc. of the 18th ACM Symposium on Virtual Reality Software and Technology (VRST2012)*, pages 201-202, Toronto, Canada, December 2012.

Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. 3d interaction assistance in virtual reality : A semantic reasoning engine for context-awareness. In *Proc. of the 1st Internationnal Conference on Context-Aware Systems and Applications (ICCASA)*, volume 109 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 30-40. Springer, Ho Chi Minh City, Vietnam, 2012.

Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. 3D Interaction Assistance Through Context-Awareness : A semantic reasoning engine for classic virtual environment. *In Proc. of the International Conference on Computer Graphics Theory and Applications (GRAPP 2012)*, pages 562-567, Rome, Italie, 2012.

Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. 3D Interaction Assistance Through Context-Awareness. In *Poster Session of the Proc. of IEEE Virtual Reality (VR 2012)*, pages 103-104, Orange County, CA, États-Unis, 2012.

Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. A Discrete Hidden Markov Models Recognition Module for Temporal Series : Application to Real-Time 3D Hand Gestures. In *Proc. of the 3rd IEEE International Conference on Image Processing Theory, Tools and Applications (IPTA 2012)*, pages 299-304, Istanbul, Turquie, 2012.

Yannick Dennemont, Guillaume Bouyer, and Malik Mallem. Réalisation d ?un module de reconnaissance de gestes continus en temps réel pour la réalité virtuelle. In *Actes des Journées de l'AFRV*, pages 95-101, Orsay, France, 2010.

REMERCIEMENTS

Je remercie mon directeur de thèse, M. Malik Mallem, qui m'a accepté tout d'abord comme étudiant en Master Réalité Virtuelle et Système Intelligent avant de continuer en thèse. Je remercie également l'ENS Cachan qui m'a permis d'explorer cette nouvelle branche pour terminer mes études. J'ai pu bénéficier d'une grande liberté quant à l'approche et la réalisation de ces travaux tout en étoffant mon bagage initial de physique appliquée dans de nombreux nouveaux domaines. Notamment concernant les environnements virtuels et l'intelligence artificielle qui, s'ils ne sont plus de la science fiction depuis longtemps, en gardent un peu l'imaginaire et l'attrait.

Je remercie également mes encadrants, M. Samir Otmane et M. Guillaume Bouyer, pour leurs conseils et leur aides dans les démarches scientifiques. Notamment M. Otmane pour les nombreuses opportunités internationales dont j'ai pu bénéficier et M. Bouyer pour les discussions qui m'ont souvent réorienté mais aussi encouragé.

Je remercie l'ensemble du laboratoire et particulièrement toute personne ayant travaillé à un moment donné dans la salle Ax14 et qui a ainsi participé à l'ambiance vraiment sympathique y régnant. Tout particulièrement le groupe de midi et ses débats diversifiés avec Mouna, Mehdi, les deux Christophe, les deux Maxime, Cyril, Jean-Clément, Jean-Yves...

Je glisse un petit mot, probablement à leur surprise, aux professeurs et élèves des clubs Ronin Ryu de Yoseikan Budo à Cachan et Kung Arts de Sanda à Combs-la-ville. La pratique sportive que j'ai adoptée via les arts martiaux à Cachan est essentielle tant pour le corps que l'esprit. Elle aura grandement favorisé ma motivation et mon inspiration pendant mes études.

Je tiens à remercier ma famille, tout particulièrement ma mère et ma soeur. Mon épanouissement personnel et professionnel leur doit depuis longtemps beaucoup. Également pour leur soutien depuis la Réunion dans cette période bien mouvementée, notamment sur le plan personnel.

Et finalement, Inès, merci d'avoir su ainsi éclairer cette dernière année de thèse.

"ti pa ti pa n'arivé"
pas à pas, nous arriverons

TABLE DES MATIÈRES

LISTE DES FIGURES

INTRODUCTION GÉNÉRALE

Ces travaux concernent l'adaptation de l'interaction 3D, la conscience du contexte et la gestion de sémantique. Nous cherchons à obtenir ces trois avantages automatiquement et de manière générique au sein d'applications décrites de manière classique. Cette approche hybride d'environnements intelligents ajoute des difficultés pour nos travaux mais permettent ainsi des études sémantiques pour des environnements qui ne le sont pas. L'expressivité et l'évolutivité de l'approche ont été favorisées afin de permettre un usage durable pour des applications a priori inconnues. Ce manuscrit est divisé en trois parties, l'état de l'art comprenant les chapitres 1 à 3, les contributions concernant les chapitres 4 à 6 et les annexes avec notamment la bibliographie.

Le premier chapitre permet de poser la problématique et d'en comprendre les enjeux. Il commence par présenter la réalité virtuelle dont l'interaction 3D est un domaine de recherche. Puis les adaptations de cette interaction 3D comme piste d'évolution sont détaillées, car environnements et tâches deviennent de plus en plus complexes et diversifiés. La notion de contexte, apparaissant de manière récurrente dans notre expression, est alors explicitée. L'usage du contexte afin d'obtenir une gestion générique et automatique des adaptations de l'interaction 3D conclut notre positionnement.

Le deuxième chapitre fait un tour d'horizon des approches rencontrées. La conscience du contexte est d'abord positionnée vis-à-vis de l'interaction adaptative, qui sont tous deux fondamentalement des visions différentes d'un même problème, et de l'intelligence artificielle dont elle est un représentant issu de l'évolution des systèmes intelligents. Puis les différentes propriétés et domaines d'applications de la conscience du contexte sont présentés. Suite à cela, certains travaux pour l'interaction 3D, l'informatique ubiquitaire et les environnements sémantiques complets sont détaillés. Notre approche est alors positionnée en faveur d'une modélisation nouvelle pour obtenir évolutivité et expressivité en s'inspirant des avantages de chaque méthode.

Le troisième chapitre s'intéresse alors aux différentes modélisations et formalisations rencontrées. Tout d'abord les modèles issus de la conscience du contexte même, mais également ceux venant de l'étude des adaptations et enfin les formalismes de représentation et de raisonnement permettant d'implémenter ces modèles. Notre choix est alors effectué et porte sur les graphes conceptuels au sein d'un raisonnement sémantique.

Le quatrième chapitre traite de la création de notre raisonnement sémantique externe, expressif et modifiable. La place du moteur, ses fonctionnalités et son usage pratique conceptualisé sont premièrement globalement présentés. Puis les formalismes et leurs propriétés, notamment via notre usage au sein de la plate-

forme d'implémentation, sont présentés : l'ontologie, les graphes conceptuels et l'interpréteur de raisonnement sémantique. La création de notre méta-interpréteur permettant la gestion du temps et de la confiance, afin de gérer une interaction 3D dynamique et une comparaison de réactions à la sémantique différente, est introduite. A l'aide de celui-ci, le processus de réactions automatique gérant les moyens propres aux applications et l'impact des réactions est détaillé. Enfin, des concepts généraux formant une extension des capacités premières du moteur ainsi formé sont également présentés.

Le cinquième chapitre détaille la création des outils et des règles permettant d'obtenir un comportement propre, à savoir nos assistances adaptatives à l'interaction 3D. Un résumé des lignes directrices de ces créations est d'abord présenté. Puis sont abordés les capteurs permettant d'extraire des informations de nos environnements classiques, notamment les indices sur l'intention de l'utilisateur. Des interprétations via des combinaisons d'informations, comme des jugements de valeur sur la situation, permettent alors d'effectuer des étapes générales et extensives de notre raisonnement. A l'issue de la création d'effecteurs, des assistances peuvent alors être gérées se basant sur ces étapes et informations. Un ensemble de raisonnements pratiques et de possibilités de réification ont ainsi été créés et prêts à l'emploi.

Le sixième chapitre illustre des usages du moteur. Premièrement des applications en ligne, avec une impression de temps réel grâce à la transparence du raisonnement automatique, dans un environnement qui a permis des tests tout au long de l'évolution du moteur. Il permet notamment d'illustrer les possibilités d'identification des intérêts et difficultés de l'utilisateur ainsi que des premières assistances. Puis un environnement pour une évaluation envisagée (mise de côté à cause du temps de réponse élevé) permet d'obtenir des comportements d'assistances en fonction de la situation relevée d'un objectif connu. Enfin une étude hors ligne des étapes internes menant au choix de la technique d'interaction montre l'accès possible au raisonnement du moteur et son usage possible pour une analyse de l'activité de sessions enregistrées.

Enfin une conclusion et nos perspectives à court et à long termes sont présentées, avant la dernière partie contenant les annexes. La sémantique intégrée et l'usage de graphe conceptuels sont promis à un usage de plus en plus fréquent, dont notre moteur permet une étape intermédiaire pour les applications classiques.

Première partie

ÉTAT DE L'ART

"I fo apiye si pié gos pou lève pié droit'"
Il faut s'appuyer sur le pied gauche pour lever le pied droit

1

PROBLÉMATIQUE ET OBJET DE CETTE THÈSE

Résumé : *Les tâches dans les environnements virtuels immersifs sont associées à des techniques d'interaction 3D et à des dispositifs. Alors que les environnements et les tâches deviennent de plus en plus complexes, les techniques ne peuvent plus être les mêmes pour chaque application, voire pour les différentes situations au sein d'une application. Une solution peut être d'adapter l'interaction en fonction des besoins. Ces adaptations peuvent être effectuées manuellement ou automatiquement, créant alors une interaction adaptative. La formalisation d'une telle assistance automatique nécessite la gestion d'informations pertinentes au vue de la situation : le contexte de l'interaction. Cependant le contexte est relatif et dépend ainsi des applications, du temps, de l'utilisateur etc. Il n'est pas toujours pleinement prévisible et émerge de l'activité. Il faut donc pouvoir obtenir une gestion du contexte générique, utilisable, extensive et modifiable dynamiquement qui soit également facile à intégrer par une majorité d'applications*

1.1 INTRODUCTION À LA RÉALITÉ VIRTUELLE

La réalité virtuelle est un terme porteur de nombreuses promesses. En effet, l'être humain n'a de cesse d'user de représentations, qu'elles aient pour finalité la réflexion, l'expression artistique, ludique, etc. Cette représentation peut se faire à l'aide de nombreux médias (écriture, dessin, films, théâtre etc.). La réalité virtuelle est une des ces possibilités d'expressions, qui semble pouvoir répondre à toutes les attentes, en permettant la création d'un monde spécifique. Dans cette section sont présentés les différentes définitions et concepts de ce domaine.

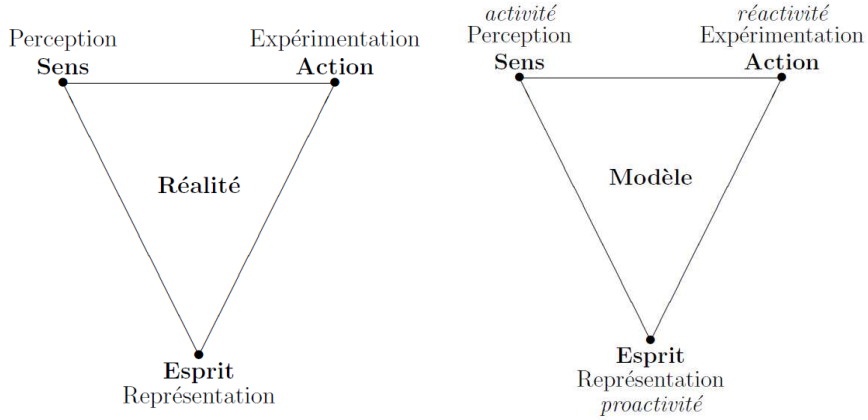
1.1.1 *Définitions de la réalité virtuelle*

La *réalité virtuelle* est un terme consacré, choisi initialement par analogie avec le terme anglais *virtual reality*. Cette dénomination frappe l'imagination en désignant ainsi par l'adjectif *virtuelle*, c'est-à-dire qui n'est qu'en puissance, une *réalité*, fortement liée au monde concret. Le terme n'est pas un oxymore même s'il en a l'apparence de prime abord, car la réalité comprend également ce qui est en puissance, le virtuel, qui ne s'oppose en fait qu'à ce qui est actuel. Cependant une traduction probablement plus juste du terme anglais initial serait une *quasi-réalité*.

Définir la réalité n'est également pas chose facile. Mais quel que soit son statut, trois médiations peuvent intervenir entre elle et l'homme [87], en tant que trois aspects d'une même réalité (Fig. 1a). Ainsi nous pouvons avoir accès à la réalité à travers nos sens (le monde perçu), nos actions (le monde expérimenté) et notre

esprit (le monde imaginé). Obtenir une *quasi-réalité* doit donc permettre d'obtenir ces trois médiations : elle doit être alors perceptible, interactive, et permettre de se forger une représentation mentale.

La notion de virtuel associée à cette réalité peut être rapprochée de son usage en physique. Elle désigne alors une conception spécifique du monde permettant sa modélisation. Ainsi les modèles physiques sont créés à partir d'observations et ils doivent alors permettre en retour d'expliquer ces mêmes observations. Par extension, ils permettent également de prévoir les observations futures (et d'être ainsi pro-actif dans le monde). C'est pourquoi *la réalité virtuelle est virtuelle parce qu'elle concerne résolument l'univers des modèles. Une réalité virtuelle est un univers de modèles au sein duquel tout se passe comme si les modèles étaient réels parce qu'ils proposent simultanément la triple médiation des sens, de l'action et de l'esprit* [87], Fig 1b.



(a) Les trois médiations du réel (b) Un modèle offrant les trois médiations

FIGURE 1: La réalité virtuelle comme un modèle supportant les trois médiations du réel [87]

Ainsi, la réalité virtuelle permet *à une personne (ou à plusieurs) une activité sensori-motrice et cognitive dans un monde artificiel, créé numériquement, qui peut être imaginaire, symbolique ou une simulation de certains aspects du monde réel. D'un point de vue physique, l'utilisateur est relié à une ou plusieurs machines au niveau de ses sens et ses réponses motrices via des interfaces motrices et sensorielles* [5]. La réalité virtuelle est alors définie comme *un domaine scientifique pluridisciplinaire qui permet à un ou plusieurs utilisateurs en immersion pseudo-naturelle d'être en interaction temps réel avec un environnement virtuel par l'intermédiaire de canaux sensori-moteurs* [5]. L'utilisateur est donc au centre des problématiques de la réalité virtuelle. Celle-ci touche ainsi à la fois au domaine des sciences et techniques mais aussi à celui des sciences humaines afin de penser et réaliser une expérience de l'utilisateur.

Le type d'expérience de l'utilisateur propre à la réalité virtuelle peut être mis en valeur et quantifié selon trois axes : l'autonomie, l'immersion et l'interaction (Fig. 2). Un virus informatique privilégie grandement l'autonomie, le cinéma 3D l'immersion et le jeu vidéo l'interaction. La réalité virtuelle doit ainsi être capable de réussir simultanément selon ses trois axes. La combinaison de l'immersion et de l'interaction permet de quantifier la présence, qui reflète à quel point l'utilisateur se sent intégré à l'environnement.

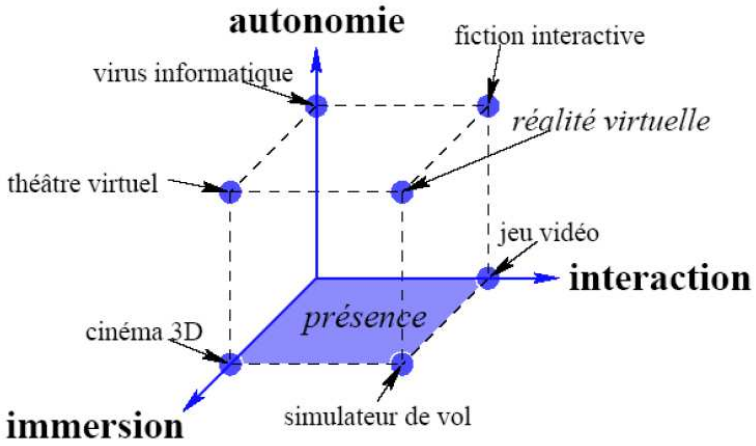


FIGURE 2: Concept de présence et axes définissant la réalité virtuelle [88]

La conception d'un dispositif de réalité virtuelle doit donc gérer des immersions et l'interactions (I2) à plusieurs niveaux [5], Fig. 3 :

- Au niveau d'I2 sensori-motrices sont définies les liaisons physiques entre la machine et l'Homme.
- Au niveau d'I2 cognitives sont définies les processus cognitifs qui rentrent en jeu et le logiciel de réalité virtuelle. L'ensemble doit permettre de rendre invisible à l'utilisateur le niveau I2 Sensori-motrices.
- Au niveau d'I2 fonctionnelles sont définies les activités de l'Homme dans l'environnement virtuel. Ce niveau traite donc de l'application en elle-même et de ses objectifs.
- Un quatrième niveau d'I2 Sociales peut être introduit, y sont définies les aspects de collaboration entre utilisateurs.

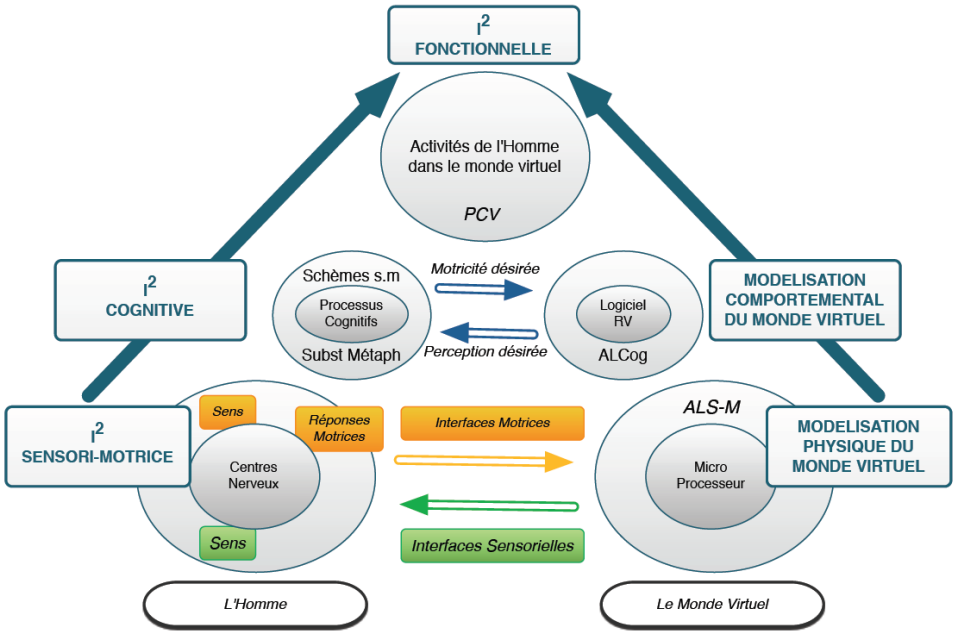


FIGURE 3: Différents niveaux de l'interaction entre un utilisateur et un environnement virtuel [5], version illustrée par [10]

Enfin les environnements virtuels peuvent n'occulter que partiellement la réalité. Cela est un défaut lorsque la réalité perceptible réduit l'immersion, par exemple

dans le cas de limitations technologiques (e.g un angle de vision insuffisant). Mais cela peut être une volonté : intégrer un environnement virtuel au monde réel est le domaine de la réalité augmentée. Ce mélange de réalité et de virtualité peut par exemple être obtenu via un casque semi-transparent [1]. Ainsi il existe un continuum entre environnement virtuel et environnement réel : la réalité mixte [58], Fig 4. La réalité augmentée y est définie comme gardant une proportion plus grande d'éléments du réel tandis que la virtualité augmentée, elle, propose plutôt des éléments virtuels. S'il est possible de tendre à l'extrême vers la réalité (en ne lui apportant aucune modification), la réalité virtuelle achevée est un idéal qui suppose que l'on ne perçoivent plus d'éléments du monde réel. L'ensemble de ces travaux relève donc en pratique de la réalité mixte.

FIGURE 4: Continuum de la réalité mixte, réinterprétation [58]

1.1.2 *L'interaction 3D*

La réalité virtuelle est donc notamment interactive. Cette interaction avec un environnement virtuel en trois dimensions [2] est effectuée à travers des dispositifs et à l'aide de techniques d'interaction. Les dispositifs tout comme les techniques peuvent être extrêmement variés, à l'image des applications de réalité virtuelle. D'ailleurs l'interaction 3D, virtuelle, n'a pas à respecter les contraintes d'un monde réel (le déplacement par téléportation est ainsi envisageable). Néanmoins, l'interaction 3D tend souvent à imiter les schèmes comportementaux naturels afin de rendre l'interaction plus facile et plus immersive. Un exemple est de mimer la préhension (la technique d'interaction) à l'aide de gants de données (le dispositif matériel) pour déclencher la sélection d'un objet. La même idée est déjà utilisée dans le cadre du clic de souris dans une interface à icônes classique.

Les interactions 3D sont ainsi très variées, mais peuvent être classifiées en tâches élémentaires [13] :

- la sélection : la désignation d'un objet ou d'un ensemble d'objets.
- la manipulation : la modification des propriétés des objets. Cette tâche est très proche de la sélection car une cible doit être au préalable désignée ;
- la navigation : la recherche d'itinéraire et le déplacement dans l'espace.
- le contrôle d'application : les communications directes avec l'application, par exemple afin de changer manuellement de tâche.

Une autre taxonomie de l'interaction 3D est effectuée selon le comportement de l'utilisateur, séparé en Primitives Comportementales Virtuelles [5] qui sont :

1. Comme tout récemment les lunettes Google [93], qui vise à démocratiser la réalité mixte comme la Wiimote [91] puis Kinect [92] l'ont fait pour l'interaction gestuelle.
2. La dimension n'est pas le critère majeur de la réalité virtuelle. Mais se contenter de deux dimensions réduit le potentiel d'immersion de l'environnement. Notre médiation avec le monde réel, nos schèmes comportementaux naturels nous ont habitués à un monde 3D.

– Observer le monde virtuel ;
– Se déplacer dans le monde virtuel ;
– Agir dans le monde virtuel ;
– Communiquer avec autrui ou avec l'application.

Ces taxonomies sont interdépendantes : observer le monde virtuel est un comportement qui se retrouve à la fois dans une tâche de sélection ou de navigation. La sélection et la manipulation sont des tâches qui relèvent du comportement d'action dans le monde virtuel. L'usage des deux simultanément est enrichissante. Ainsi détecter que l'utilisateur est en phase d'observation alors que la tâche est une sélection permet par exemple de s'interroger sur des difficultés potentielles de cette sélection. Ces taxonomies permettent donc toutes deux d'aborder de manière générale l'interaction 3D.

L'interaction 3D est un des domaines majeurs de recherche pour la réalité virtuelle tant pour la création et la gestion de dispositifs que de techniques d'interaction les utilisant. Un état de l'art détaillé a été effectué par Ouramdane et al. [67].

1.2 VERS DES ADAPTATIONS DE L'INTERACTION 3D

L'interaction 3D pose des problèmes bien spécifiques à gérer dus à la multiplicité ainsi qu'à la complexité croissante des tâches et des dispositifs. Or l'exploitation des environnements virtuels peut être d'autant plus intéressante et enrichissante que cette complexité gérée est élevée. Il faut cependant que l'utilisateur puisse bénéficier de l'expérience au sein de l'environnement virtuel sans ressentir cette complexité, qui doit être la plus transparente possible. La gestion de l'ensemble des situations différentes, à des niveaux de granularité tout aussi différents, offre de nombreuses opportunités de recherche dans le domaine. Ainsi, l'interaction avec un environnement de grande taille comme la visite d'un musée virtuel ou celle nécessaire par la simulation de formation d'un chirurgien ne nécessiteront pas l'usage des mêmes techniques d'interaction. De plus, une visite sans contrainte pour le chaland virtuel ou un processus fixe de formation composé de différentes étapes ne demandent pas la même connaissance a priori des intentions et des objectifs, permettant au designer de pré-déterminer l'adéquation d'une technique à un instant donné de l'application.

1.2.1 *L'adaptation comme axe de recherche*

L'adaptation de techniques d'interactions à la situation est un moyen de résoudre ces difficultés. L'adaptation en soi est variée et possède de nombreux critères pertinents pour sa compréhension (voir section 3.2.1) notamment l'utilisateur. La personnalisation de l'interaction a été ainsi l'un des premiers motifs d'adaptation de l'interaction 3D [94]. Ainsi, Wingrave et al. personnalise la sélection d'une sphère virtuelle parmi trois, dans le cadre de l'usage d'un gant de données et du geste de pincement. La personnalisation est effectuée à l'aide de réseaux de neurones. Cette approche est robuste mais peu extensible car la représentation du réseau est alors définie dans ce cas précis, c'est-à-dire ne permettant l'existence que de trois objets.

Afin de déterminer des critères élémentaires nécessaires à la personnalisation des techniques d'interaction, Wingrave et al. ont tenté d'évaluer et d'obtenir les préconceptions que se font les utilisateurs de différentes techniques d'interaction, décrites de manière vague. L'hypothèse était qu'avec peu de retour de la part du système, les utilisateurs effectueraient alors leur tâche naturellement. L'évaluation a fait ressortir que l'utilisateur n'a pas une préconception bien définie, et que le retour sur interaction est prédominant. En l'absence de retour clair, l'utilisateur ne tend pas vers son interaction naturelle mais au contraire l'adapte afin d'adopter l'attitude qu'il pense meilleure pour le système [3]. Au final, l'interaction effectuée ne bénéficie en général ni à l'utilisateur ni au système.

En aparté, nous avons expérimenté le même phénomène lors de l'évaluation de notre module de reconnaissance de geste [34]. L'utilisateur effectuait des gestes plus rapides car ressentis comme plus énergiques afin d'aider la reconnaissance suite à des échecs précédents. La longueur du geste devenait alors en fait trop courte pour le module. L'utilisateur, en l'absence de retour, tente en effet, souvent inconsciemment, de s'adapter au système selon ses propres conceptions. Cet effet d'expérimentation et d'adaptation de l'utilisateur reste présent lors de l'usage d'un système lui même adaptatif, et à nouveau généralement au détriment des deux parties [29].

L'évaluation est alors changée et porte sur des variantes de techniques d'interaction bien définies. L'apparition d'usages privilégiés de paramétrages différents de techniques selon les propriétés de l'objet ciblé est alors notée. Wingrave et al. introduisent alors l'idée d'une interaction à nuances. Ces nuances, modifiant une technique d'interaction habituelle, seraient à appliquer par une interface douée d'une certaine forme de raison. Cette idée est aussi développée par Bowman et al. qui notent le ralentissement des innovations fondamentales en interaction 3D bien que les problématiques ne soient pas résolues [14]. Ils proposent alors des axes pour les recherches futures, notamment d'établir des techniques d'interactions spécifiques à une tâche (spécificité) ou d'adapter des techniques existantes avec l'idée de nuances personnalisées (déclinaisons).

Ces notions ont été évaluées et peuvent toutes deux améliorer les performances et le confort de l'interaction 3D [20][82]. De manière générale, de nombreux travaux ont ainsi adapté différents aspects de l'interaction 3D :

- Le changement de technique d'interaction (ou spécificité) [62][20][82] ;
- L'adaptation de la technique elle-même (ou déclinaison)[62][20] ;
- L'amélioration de la technique par des modalités [49][12][62] ;
- L'exécution automatique d'une partie de la tâche [21].

De manière pratique, les gains (souvent multiples) envisagés vis-à-vis de l'interaction sont :

- De l'accélérer [21][82] ;
- De diminuer la charge cognitive [83][49][95] ;
- De la personnaliser [94] [62] ;
- D'ajouter/de gérer des possibilités [55][12].

3. Il devient ainsi l'expérimentateur du modèle virtuel, et déduit de la réactivité de ce monde une représentation d'esprit propre afin de pouvoir être pro-actif.

1.2.2 *Adaptation et uniformisation*

L'adaptation a des avantages et des inconvénients, notamment comparée à l'approche opposée d'uniformisation (Fig 5). L'usage d'une technique d'interaction 3D uniforme permet notamment une simplicité de mise en oeuvre, un transfert d'apprentissage, une sûreté et une réutilisabilité. Celle-ci peut en revanche être limitée en pratique car ressentie comme rigide ou inadéquate pour la situation. Les adaptations de l'interaction, comme les spécificités et les déclinaisons, répondent au besoin de gérer ces différents contextes, d'assouplir la communication et d'obtenir l'efficacité et la personnalisation. Elles sont en revanche plus complexes à mettre en oeuvre et peuvent également être erronées.

	Uniformisation	**Adaptation**
Intérêts	– Référence communautaire voire culture de masse – Transfert d'apprentissage entre applications – Réutilisation et interopérabilité – Simplicité de mise en œuvre et de maintenance (économie) – "Sureté" et contrôle du processus	– Ergonomie (confort et efficacité dans l'interaction) – Interaction personnalisée et reconnaissance de l'usager – Flexibilité et souplesse d'usage – Prise en compte du contexte et de la situation d'usage
Inconvénients	– Non prise en compte des différences inter-individuelles – Décontextualisation – Rigidité	– Risque d'isolement – Complexité de mise en œuvre et d'évaluation – Effets négatifs lors d'une mauvaise adaptation

FIGURE 5: Intérêts et inconvénients des approches d'uniformisation et d'adaptation [84]

Mais il y a plusieurs plusieurs formes (et plusieurs cibles) d'adaptations. Une première distinction globale est effectuée selon l'acteur (utilisateur ou système) et le moment de l'adaptation. Ces formes ne sont pas mutuellement exclusives et plusieurs sont souvent utilisées au sein d'une même application [68]. Un système peut être :

- Modifiable, le système peut être modifié pour convenir à des besoins particuliers.
- Personnalisable, lorsque que le système permet spécifiquement la modification de paramètres et la définition de profil par l'utilisateur.
- Adaptable ajoute la possibilité au système d'effectuer l'ajustement lui-même.
- Adaptatif [4] dénote que le système soit capable de déterminer un modèle d'utilisateur et d'effectuer un lien entre le modèle et les possibilités d'ajustements en temps réel.

4. De l'anglais *adaptivity* par rapport au terme précédent *adaptability*

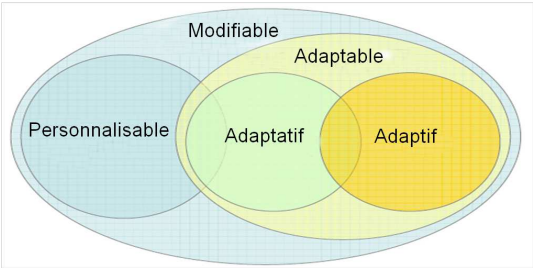


FIGURE 6: Différentes formes d'adaptation, adapté de [68]

Ainsi les adaptations précédentes peuvent être effectuées automatiquement par le système. L'interaction 3D est alors au mieux simultanément adaptive et adaptative, capable globalement de s'adapter tout en établissant un modèle utilisateur influençant son comportement de manière dynamique. Nous retiendrons que des adaptations automatiques de l'interaction 3D la rendent ainsi globalement adaptative, au vue de la situation. Une gestion générique des adaptations, notamment automatiques, doit ainsi être elle modifiable et idéalement au moins adaptable vis-à-vis de l'application d'implémentation de ces adaptations.

D'autres termes se retrouvent dans la littérature pour qualifier le processus d'adaptation des systèmes. On parle ainsi d'une adaptation réactive lorsque la réaction est définie par un ensemble d'hypothèses qui, reproduites à un instant différent, provoqueront la même réponse. A contrario une interaction pro-active possède une réaction qui s'inscrit dans un plan. Ainsi le même stimuli à des instants différents peut provoquer des réactions différentes. En effet, le système peut alors prévoir a priori les conséquences de l'adaptation, intéressante ou non selon l'objectif en cours. Ces deux qualificatifs d'adaptations concernent un système au moins adaptatif et mettent en exergue l'importance du contexte pour l'adaptation, incluant ici le temps et l'objectif.

Ces qualificatifs dénotent une différence d'intention initiale[5]. De la même manière, l'utilisateur au sein d'un environnement virtuel teste la réactivité de l'application grâce à son interaction et sa médiation d'esprit lui permet alors d'être pro-actif, de prévoir un comportement avant d'effectuer l'interaction [87] (Fig 1b). Ainsi nous parlerons plus généralement d'une adaptation pro-active comme permettant l'adaptation d'un aspect important pour une interaction avant que celle-ci n'ait commencé réellement (comme un changement de technique d'interaction au vue de la tâche à venir). Une adaptation réactive s'effectuera en cours d'interaction.[6]

1.2.3 *Interaction 3D adaptative*

De nombreux travaux ont été effectués quant à l'adaptation et l'adaptativité de l'interaction 2D notamment pour l'informatique ubiquitaire et la plasticité des interfaces. La liberté et la complexité de l'interaction ainsi que du suivi de l'utilisateur est alors moindre que dans un environnement 3D et les adaptations sont ainsi souvent facilitées, Fig 7. Des exemples et différences sont présentés dans

5. Cette notion de plan étant aussi la différence entre les capacités d'autorégulation et d'automédiation d'un processus d'adaptation [90]. D'autres concepts pour la modélisation de l'adaptation sont introduits section 3.2.1.

6. Cet usage est compatible avec la notion de plan réactif qui est un plan dont l'objectif final évolue en cours d'interaction.

le cas du web, média 2D à l'origine mais qui intègre de plus en plus d'applications 3D [26]. Ainsi certaines approches tentent de gérer les deux types d'interactions [79] au sein d'un même système en réifiant des tâches génériques (manipulation, navigation, visualisation et écriture) en fonction des possibilités de la plateforme (notamment 2D ou 3D).

	Web-based hypermedia	3D Web sites
presentation container	page	3D space
content media	mainly text, but also images, videos, ...	mainly 3D models, but also text, images, videos, ...
structural organization	graph of pages	3D space or graph of 3D spaces
navigation	through hyperlinks	by moving in 3D space (e.g., walking, flying) and teleporting; also through hyperlinks
other common users' activities	reading pages, filling forms	3D object manipulation (clicking, moving, …),

FIGURE 7: Difficultés supplémentaires de l'interaction 3D pour le web [26]

Parmi les travaux d'interaction 3D adaptative, il y a deux approches : celle d'une adaptation intégrée directement à l'interaction et celle d'un processus externe la modifiant. Les approches intégrées nécessitent en général peu d'informations différentes. Un des exemples les plus connus est probablement la technique de sélection Go-Go [70]. La position relative de la main de l'utilisateur provoque un déplacement non linéaire d'une main virtuelle, assistant automatiquement l'utilisateur pour une sélection distante. Un autre exemple est la technique Fly over [11] qui propose à partir de la même information initiale une détection de la tâche primaire (sélection ou navigation). La vitesse et le type de déplacement (une combinaison de rotation et de translation) sont alors adaptés automatiquement en fonction de cette tâche. Une autre technique permet également une transition continue entre tâches [27]. Elle permet de distinguer la volonté de sélection-déplacement et celle de sélection multiple par l'analyse de la forme du geste d'approche effectué. Dans tous ces cas, l'adaptativité se confond avec la technique elle-même. L'adaptativité est néanmoins reproductible par un système externe disposant de la même information et des moyens de modification de l'interaction.

Enfin, des approches extériorisent la capacité d'adaptation. Par exemple le framework M3I (multimodal manipulation for 3D Interaction) [49] permet de relever une ambiguïté ontologique d'une commande d'entrée, par exemple vocale. Ainsi lorsque l'utilisateur donne une commande comme un classique[7] "met la lampe ici" . Pour trouver la lampe concernée, le système considère tout d'abord le dernier manipulé de ce nom. Autrement, c'est alors celui présent dans le champ visuel de l'utilisateur et sinon le plus proche. L'objet est alors déposé sur l'objet le plus proche du pointeur de l'utilisateur. Ainsi, l'approche doit donc pour cela gérer une partie du contexte et une description sémantique de la scène.

7. Une partie des problématiques était déjà présente dans "put-that-there" [8]. La gestion de l'ambiguïté était principalement faite via le contexte de localisation : *ce qu'"est" un objet est alors en quelque sorte "où" est l'objet.*

Mais l'adaptation peut concerner bien d'autres choses que l'interprétation d'entrées utilisateur et peut ainsi globalement affecter l'environnement ou la technique d'interaction. Des approches d'adaptations externes à volonté générique sont ainsi détaillées dans le chapitre suivant (section 2.3 et 2.4). Parmi ceux-ci, les adaptations de l'interaction 3D sont obtenues par la création d'environnement sémantique intelligent [56], de supervision multimodale [12], de gestion de conscience du contexte pour la réalité mixte [55], de personnalisation active et prototypage rapide d'adaptations en général [61] (qui se place dans la lignée des travaux de Wingrave et al.) ou ciblant particulièrement les environnements de formation [41].

La détermination des informations pertinentes au vu de chaque situation est alors souvent à établir pour obtenir des adaptations. Ainsi le superviseur de rendu adaptatif [12] choisit des modalités et leur attributs sur chaque canal sensoriel en se basant sur une description de l'interaction à l'aide de la tache, un ensemble de paramètres la concernant (e.g la présence de contrainte, la précision voulue etc.) ainsi que le type de donnée (objet, zone etc.) et des caractéristiques (statique ou non, unique ou non etc.). Cette approche est plus détaillée section 2.3.1. Frees propose l'association de paire tâche/contexte afin d'obtenir des adaptations cohérentes plus facilement (Fig. 8). Le contexte pertinent est similaire avec la notion de degré de contrôle (l'équilibre entre précision et rapidité), la portion de l'espace principal de travail, l'objet d'intérêt principal, la notion de groupe ou de relation éventuelle entre objet et enfin la présence de contraintes.

	Level of Control	Workspace	Frame of Reference	Grouping	Constraints
Selection	X	X		X	X
Attach Object				X	
Perform Translation	X	X	X	X	X
Perform Rotation	X	X	X	X	X
Perform Scaling	X	X	X	X	X
Object Placement				X	X
Navigation	X	X	X		X
Viewpoint Control	X	X	X		

FIGURE 8: Pertinence d'éléments de contexte selon l'activité [40]

1.3 NOTION DE CONTEXTE

La notion de contexte est ainsi invoquée régulièrement dans les sections précédentes ; parfois explicitement comme critère en faveur de l'adaptation ou afin de qualifier généralement le contenu d'une situation ; parfois implicitement en parlant d'informations pertinentes. La section suivante présente plus en avant les définitions du contexte et ses propriétés. Le contexte est fondamental pour la gestion d'adaptations.

1.3.1 *Définitions du contexte*

Premièrement il existe plusieurs définitions du contexte dans le cadre de l'interaction homme-machine. Une définition formelle et reconnue de ce contexte est [36] : *Le contexte est l'ensemble des informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité peut être une personne, un lieu ou un objet considéré comme pertinent par rapport à l'interaction entre un utilisateur et une application, incluant l'utilisateur et l'application elle-même.*

Une autre définition [4] est que *le contexte agit comme l'ensemble des contraintes qui influencent le comportement d'un système (utilisateur ou ordinateur) engagé dans une tâche donnée.* Bazire and Brézillon analysent par ailleurs un corpus de définitions du contexte issu de différentes disciplines et montrent sa grande diversité.

Et ainsi, issue des sciences sociales mais revendiquée comme devant affecter également fondamentalement le design d'applications, une autre définition réfute l'existence du contexte en soi. En effet, le contexte est alors défini comme une *propriété émergente de l'activité* [37], qui est ainsi activement produit, maintenu et adopté au cours de celle-ci. Le contexte n'apparait donc réellement qu'au sein de chaque instance particulière.

Ces définitions sont influencées par deux vues fondamentalement différentes de l'expérience en sciences sociales [37]. L'approche positiviste décrit par analogie scientifique les phénomènes sociaux comme explicables par un modèle objectif capable de représenter les causes sous-jacentes et permet ainsi un traitement quantitatif. Au contraire, l'approche phénoménologique rejette l'idée qu'il y ait une telle réalité objective impliquée. Les phénomènes sociaux sont alors des propriétés émergeant des interactions. Ainsi le monde perçu n'est pas une réalité absolue mais essentiellement issu d'un consensus d'interprétation. Les catégories sont imposées au monde par notre interaction et non pas existantes en soi [8]. Ainsi dans les deux premières définitions, d'approche positiviste, un ensemble d'informations ou de contraintes est considéré comme pertinent ou influant vis-à vis de l'interaction ou de la tâche. La troisième, phénoménologique, n'oppose pas le contenu (l'activité, l'interaction ou la tâche) comme étant défini séparément d'un contexte.

1.3.2 *La négociation du contexte pour l'interaction 3D*

Quelle que soit la définition considérée, le contexte reste d'une importance critique pour comprendre et spécifier les activités et les informations. Mais l'approche phénoménologique implique qu'il reste par nature constamment négocié et redéfini. Les premières définitions sont utiles pour se représenter le contexte. Mais il faut garder à l'esprit que cette représentation n'est en fait qu'un instantané du contexte

8. L'ethnométhodologie, traditionnellement ancrée dans la phénoménologie, spécifie que la notion d'ordinaire [76] émerge de l'interaction. C'est une catégorie imposée à une situation au sein d'une conversation et qui n'est pas innée. Elle est dépendante de la culture, du groupe social, des interlocuteurs, etc. mais plus encore est un achèvement négocié, un consensus d'interprétation qui dépend de chaque conversation. L'ordinaire n'est pas en soi. Le parallèle permet de mieux comprendre le principe phénoménologique qui est appliqué de manière similaire au contexte [37].

qui dépend de chaque instance particulière.

Ainsi dans l'expérience tentant d'évaluer les composantes élémentaires de l'adaptation [94], ces composantes restent relatives à cette même expérience. Elle permettent néanmoins de comprendre et d'assister l'activité. Cela illustre également que le contexte nait de l'interaction. En effet, sans avoir été informé des détails des techniques, et donc sans avoir négocié le contexte au préalable, l'utilisateur se forge sa propre opinion des informations pertinentes au cours de l'interaction. Son opinion dépend des retours de l'application (qui étaient alors réduits volontairement) qui permettent à l'utilisateur d'établir sa compréhension. Ces retours sont le moyen qu'a l'utilisateur d'accéder indirectement au contexte du système (ici, le fonctionnement précis supposé de la technique d'interaction). L'utilisateur adapte alors son interaction car il sait que seule sa connaissance du contexte est modifiable [9].

Comme suggéré en conclusion de l'évaluation, l'affordance est une piste de recherche [94]. L'affordance est *l'ensemble des possibilités d'action entre un environnement et un acteur* [43] . L'affordance des objets est constitutive de ces derniers bien qu'également une propriété relative, dépendant notamment de l'utilisateur qui considère l'objet. Ainsi l'affordance d'un objet est utilisée en ergonomie comme *sa capacité à suggérer son propre usage*. Elle est propre à l'objet et le designer ne modifie en fait généralement que des affordance perçues, à travers la gestion des retours du système [10] [60]. Les affordances font partie du contexte du système et sont négociées par l'utilisateur à l'aide des affordances perçues. Ainsi il est possible que l'utilisateur se méprenne sur le contexte de l'interaction, notamment les affordances du monde virtuel, lorsque cette négociation est limitée, lorsque les affordances perçues ne sont pas adaptées ou même présentes.

Jusqu'alors cette négociation définissant le contexte n'est en fait qu'une communication à sens unique. Seul l'utilisateur redéfinit sa connaissance du contexte pour s'adapter au système. De plus, il n'y n'accède qu'indirectement au cours de l'interaction. Que le système puisse directement exposer, mettre-à-jour, et permettre la modification par l'utilisateur comme par l'application de ses connaissances propres permettrait en revanche une vraie négociation, et ainsi l'émergence du contexte réel de l'interaction.

La réalisation d'un tel contexte, activement maintenu et redéfini par les intervenants, a de fortes similitudes avec la modélisation énactive, dont elle partage les racines phénoménologiques. Les hypothèses d'un modèle énactif sont [89] :

- La modélisation provient d'une praxis [11] humaine ;
- Les phénomènes sont modélisés en tant qu'entités autonomes ;
- Les interactions entre modèles passent par un milieu créé et façonné par l'activité des modèles eux-mêmes ;

9. Et cela fait encore partie du contexte. Si au cours de son interaction, le système semble d'adapter, le modèle mental de l'utilisateur s'adapte. Le contexte commun entre lui et le système a été renégocié.

10. Les affordances perçues peuvent être modifiées indépendamment des affordances réelles et inversement. Par exemple, en utilisant une interface d'ordinateur classique, bien que l'affordance d'un clic de souris soit disponible en permanence, les icônes suggèrent un usage particulier et représentent une affordance perçue. Bloquer le bouton de la souris lorsque l'on ne peut pas cliquer sur une zone modifie l'affordance réelle.

11. action avec intention

L'énaction au sein des environnement virtuels met en jeu des entités autonomes dotées à leur conception de capacités d'actions, d'adaptations et de perceptions. Elles ont des actions réfléchies qui sont des projections de la praxis du concepteur originel [89]. Ce principe peut être appliqué a la modélisation de la connaissance au sein de l'environnement virtuel. Ainsi celle-ci est relevée, construite et utilisée au cours de l'interaction (Fig.9). L'expérience de l'utilisateur doit pouvoir être enrichie de l'usage de ces connaissances qui elles-même évoluent à travers l'interaction de l'utilisateur. *Cette interaction spécifique n'a de signification que dans l'expérience de chacun* [86] ; tout comme le contexte émerge de chaque instance et d'un consensus d'interprétation des intervenants.

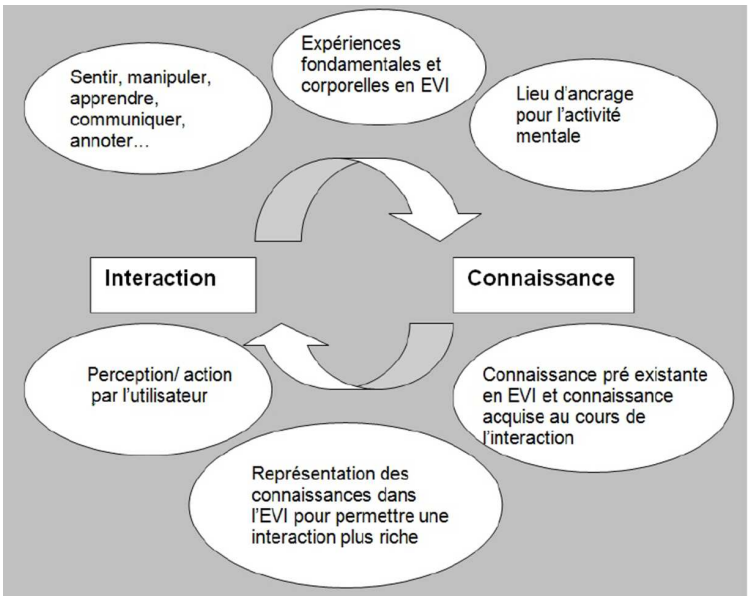


FIGURE 9: L'interaction avec les environnements virtuels informés (EVI) possédant une modélisation énactive des connaissances [86]

La modélisation idéale du contexte doit permettre d'obtenir de nouvelles informations, d'agir sur l'environnement et d'évoluer (ce qui constitue une conscience du contexte, présentée dans le chapitre suivant). Un tel modèle du contexte, capable de s'adapter par l'interaction peut permettre l'accès, l'échange et la modification de ses informations. Les différents intervenants, l'application comme l'utilisateur, peuvent ainsi directement et dynamiquement négocier le contexte avec ce modèle. La détermination d'adaptations de l'interaction adéquates est ainsi grandement favorisée. Le vrai contexte, issu de ce consensus, est alors connu du système. Celui-ci peut raisonner et fournir des adaptations automatiquement en réduisant ses erreurs. Cette négociation directe permet probablement également de réduire le manque de confiance naturel des utilisateurs vis-à-vis des processus adaptatifs [29].

1.3.3 *Propriétés du contexte*

La vision ingénierie du contexte est ainsi d'utiliser le contexte pour représenter et raisonner à propos d'un espace d'états finis dans lequel le problème est soluble. La vision cognitive est que le contexte est utilisé pour modeler les interactions et les situations dans un monde d'étendue infinie et la dimension humaine est la clef pour l'extraction d'un modèle [17]. Cependant impliquer la notion de contexte

émergeant de l'interaction comme élément de design des systèmes est revendiqué explicitement [38].

Ainsi dans la première vision, le contexte peut être perçu comme un problème de représentation [37]. Alors, le contexte :
- est une forme d'information, qui peut être connue (et donc représentée) ;
- est délimitable, notamment à l'avance pour une application donnée ;
- est stable entre différentes instances d'une même activité. La pertinence d'une information peut être déterminée une fois pour toute ;
- est séparable de l'activité.

Au contraire, le contexte peut être perçu comme un problème d'interaction et mène alors à des propriétés bien différentes [37]. Ainsi le contexte :
- est une propriété relative apparaissant entre différents objets ou activités ; ce n'est pas un ensemble d'information. Une information peut néanmoins être plus ou moins pertinente selon ce contexte ;
- possède une étendue définie dynamiquement ;
- est particulier à chaque instance d'une activité ou d'une action ;
- n'est pas séparable de l'activité : le contexte au contraire en émerge.

D'autres propriétés sont issues de l'intelligence artificielle [57]. Ainsi la formalisation du contexte de McCarthy and Sasa ne pose pas sa définition mais met en exergue ses propriétés. Le contexte est ainsi modélisé comme une classe dans une logique non monotone [12]. Cette classe permet d'expliciter la vérité d'une proposition P, relative à un contexte C, qui s'écrit alors $ist(C, P)$. Par la suite C est lui même vrai, si pour un contexte C' il existe $ist(C', C)$. Cette modélisation met en lumière que :
- un contexte est toujours relatif à un autre contexte ;
- la relativité de ces contextes a une profondeur infinie ;
- le contexte ne peut pas être décrit complètement ;
- il existe un contexte commun qui, une fois connu, permet de ne plus avoir à expliciter autant d'informations. Dans la modélisation, lorsque plusieurs contextes sont évoqués, il y a au dessus un contexte commun dans lequel les termes et prédicats précédents peuvent être retirés.

La représentation du contexte est donc utile mais intrinsèquement imparfaite. En effet, il n'est même tout simplement pas définissable en soi. Néanmoins, un ensemble d'informations servant de base d'adaptation doit respecter idéalement les propriétés du contexte : il doit être géré de manière dynamique, doit être négociable de manière implicite ou explicite entre les intervenants, pouvoir être relatif (à un autre contexte) et extensif. Néanmoins l'analyse de corpus de définitions [4] fait ressortir également que le contexte peut souvent être spécifié pour une situation en répondant aux questions : qui ? quoi ? où ? quand ? pourquoi ? et comment ? (questions quintiliennes dont la traduction anglaise abrégée,W5H, sera utilisée). *Qui* indique le sujet, l'agent d'une action. *Quoi* représente l'objet ou le sujet qui subit l'action. *Où* et *Quand* donnent les informations sur la location spatio-temporelle.

12. La logique non monotone est une logique où l'ajout d'une information peut réduire le nombre de déductions. C'est le cas de la logique du premier ordre qui peut faire de l'absence d'une information un critère de déduction.

Pourquoi donne les intentions, le but (et éventuellement les émotions) du sujet. Enfin *Comment* permet de rendre explicite la procédure pour réaliser l'action.

1.4 POSITIONNEMENT DE NOTRE APPROCHE : VERS UNE GESTION GÉNÉRIQUE DE L'INTERACTION 3D ADAPTATIVE

L'adaptation est ainsi une piste de recherche prometteuse pour le domaine de l'interaction 3D. Il est possible d'effectuer cette adaptation manuellement, par un choix préalable du concepteur ou directement décidé par l'utilisateur. Mais il est également possible de gérer ce choix par le système afin d'obtenir une adaptation automatique. L'interaction 3D devient alors adaptative. Que celui-soit explicitement défini ou implicitement utilisé, les adaptations reposent sur un contexte à déterminer en cours d'interaction.

Ainsi, afin de dépasser l'interaction figée classique, comme illustré figure 10, les systèmes adaptatifs peuvent commencer par permettre la reconnaissance d'informations de haut niveau comme les gestes ou la tâche (l'étape de détection de l'activité). Mais afin d'obtenir une meilleure adaptativité, il est nécessaire d'obtenir plus d'informations (comme le W5H) concernant l'ensemble des entités. Toutes ces informations constituent en pratique le contexte d'une instance d'interaction et évoluent dynamiquement. A l'aide de ce contexte, une interaction adaptative peut être forgée.

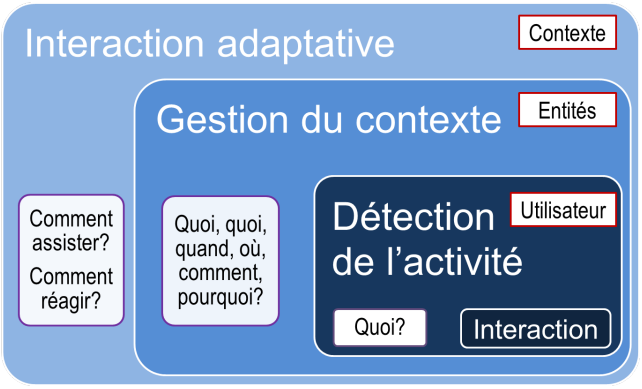


FIGURE 10: Différents niveaux afin d'atteindre l'interaction adaptative

L'objectif de cette thèse est de permettre l'exécution automatique d'adaptations pour une application quelconque de réalité virtuelle. L'interaction 3D adaptative obtenue permet alors de gérer une assistance de tout instant. Pour cela, nous avons besoin de proposer une approche générale du concept d'adaptations de l'interaction 3D et de leurs déclenchements. La gestion des informations pertinentes et des déductions qui en sont tirées doit donc être souple et expressive. L'ensemble est en fait le contexte de l'interaction. Il doit être notamment utilisable (pour pouvoir être négocié, c'est-à-dire être accessible, compréhensible et modifiable idéalement par l'utilisateur directement), extensif et modifiable dynamiquement.

Ainsi il nous faut obtenir une gestion externe aux applications de l'interaction 3D adaptative. Cette gestion doit elle-même être adaptable tant pour gérer la nature changeante du contexte que pour permettre le changement d'application communicant avec elle. Cela revient finalement à vouloir une uniformisation du processus d'adaptation afin d'obtenir à la fois les avantages de l'uniformisation et de

l'adaptation [13] (Fig 5). Ainsi l'uniformisation du processus même permet d'obtenir pour le designer une simplification de la mise en oeuvre et de la maintenance, la réutilisabilité et l'interopérabilité, le contrôle sûr, le transfert d'apprentissage, ainsi que la gestion de plusieurs références communautaires simultanément en ce qui concerne la gestion des adaptations. L'interaction 3D étant alors adaptée apporte à l'utilisateur une meilleure ergonomie, la personnalisation, la flexibilité ainsi que la prise en compte du contexte.

Finalement, les utilisateurs bénéficieront d'une assistance à l'interaction 3D, à travers un support automatique de modalités, de choix de techniques d'interaction ainsi que d'aides spécifiques à chaque application, dépendante de la situation actuelle. De plus les designers, pourront réutiliser, réarranger et modifier l'interaction 3D adaptative afin de partager des informations, de réemployer des raisonnements ou de créer des aides spécifiques pour les applications. La généricité permet également au designer de ne pas avoir à prévoir l'ensemble des situations qui peuvent être déductibles de connaissances précédentes notamment avec une gestion d'incertain.

Dans le chapitre suivant, nous nous intéresserons aux travaux permettant d'obtenir une interaction 3D adaptative par un processus externe. Dans ce cadre, le contexte, et idéalement la gestion de ses différentes propriétés, est primordial pour obtenir un processus générique. Ainsi nous nous intéresserons plus généralement aux possibilités de réactions vis-à-vis du contexte : la conscience du contexte.

13. Il faudra prendre garde en revanche à ne pas cumuler les inconvénients de chaque approche

2

LA CONSCIENCE DU CONTEXTE

Résumé : *Un raisonnement expressif, facile à utiliser et à modifier, est souhaité pour l'aide à l'interaction 3D dans les environnements virtuels. Les interactions 3D adaptatives ne sont pas nouvelles. Mais explicitant ou non le contexte à des degrés divers, elles relèvent finalement plus généralement de la conscience du contexte. De nombreux domaines autre que l'interaction 3D ont des besoins de réactions au contexte similaires, notamment l'informatique ubiquitaire. De nombreux travaux peuvent ainsi fournir une base pour notre réflexion. Cependant bien que notre problématique ait de nombreux points communs avec ces approches, aucune ne permet de répondre à l'ensemble de nos contraintes souhaitées : une assistance générique, dans un environnement qui n'est pas forcément sémantique et permettant l'ajout, la compréhension et la modification dynamique du contexte, incluant le raisonnement appliqué.*

2.1 POSITIONNEMENT DE LA CONSCIENCE DU CONTEXTE

2.1.1 *Conscience du contexte et interaction adaptative*

La gestion du contexte est nécessaire pour obtenir une interaction 3D adaptative générique. Mais globalement, de nombreux systèmes gèrent le contexte et y réagissent. Ainsi *un système est conscient du contexte s'il utilise le contexte pour fournir des informations et/ou des services pertinents à l'utilisateur, où la pertinence dépend de la tâche de l'utilisateur* [36].

Cette définition vise à réunir un ensemble de termes synonyme ou proches : adaptatif , réactif , répondant, situé, sensible au contexte et orienté environnement. Il n'est donc pas étonnant que nous ressentions un fort lien entre l'interaction adaptative et la conscience du contexte. L'interaction est ainsi au coeur même de la définition du contexte : elle permet de définir l'information de contexte, voire est siège de son émergence. De plus, la notion de conscience du contexte contient non seulement la gestion des informations pertinentes mais également celles des réactions à cette situation. Dès lors, si être adaptatif, c'est gérer les tenants et aboutissants du processus d'adaptation, la conscience du contexte est définie de même vis-à-vis de la réaction au contexte.

Ainsi si l'angle de vue est quelque peu différent, il n'y a pas de réelles différences entre l'interaction adaptative et l'interaction consciente du contexte. L'adaptation peut être vue comme une des formes de réactions au contexte. Mais, il est également discutable, selon les cibles de l'adaptation (l'information, le service, l'application

etc.), que toute réaction soit une forme ou une étape d'adaptation. Par la suite nous utiliserons de manière privilégiée la notion de conscience du contexte.

De plus, les réactions de la conscience du contexte doivent être pertinentes vis-à-vis de l'utilisateur. Et donc permettre une forme d'assistance. L'assistance à l'interaction, l'interaction adaptative et la conscience du contexte sont donc également toutes des notions très proches, voire redondantes. La distinction n'est en fait due qu'aux différentes emphases des approches. Pour la notre, l'assistance à l'interaction 3D est notre objectif et la conscience du contexte est le moyen d'y parvenir.

2.1.2 *Lien avec l'intelligence artificielle*

La conscience du contexte nécessite souvent des outils du domaine de l'intelligence artificielle [1]. Celle-ci connait de nombreuses définitions [75], Fig. 11, qui oscillent entre penser et/ou agir, de manière rationnelle et/ou humaine. Elles peuvent être illustrées de manière simplifiée (Fig. 12). L'intelligence artificielle est alors uniquement le processus de pensée, correspondant au processus interne, ou l'ensemble des processus incluant le comportement.

Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think ... *machines with minds,* in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning .. ." (Hellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et at,* 1998) "Al ...is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

FIGURE 11: Définitions de l'intelligence artificielle [75]

La gestion et la compréhension du contexte font donc appel au domaine du raisonnement et de la représentation des connaissances de l'intelligence artificielle (voir également chapitre 3). À partir de capteurs (des entrées du système qui sont représentées Fig. 12 de manière anthropomorphique), le processus interne doit prendre une décision dépendant de sa compréhension de la situation. Pour cela il dispose d'une base de connaissance ainsi que de capacités de raisonnement. Ainsi, la conscience du contexte est la mise en pratique d'une intelligence artificielle, spécialisée dans l'interaction. C'est donc un domaine multidisciplinaire qui nécessite l'usage de l'informatique pour la réalisation des applications, d'instrumentation pour avoir accès au contexte, de sciences cognitives pour optimiser la prise de décision etc.

1. Plus de détails peuvent être obtenus dans le livre de référence [75]. Une formation en ligne parcoure les chapitres de ce livre tout en proposant des exercices et évaluations [74].

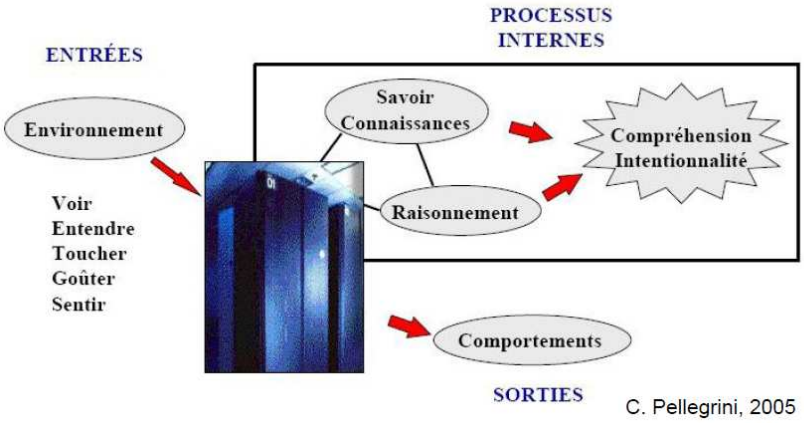


FIGURE 12: Illustration simplifiée du concept d'intelligence artificielle [54]

2.1.3 *Évolution des systèmes intelligents vers la conscience du contexte*

La conscience du contexte est issue d'une évolution des systèmes intelligents, et a souvent été présente bien que mentionnée différemment. Ainsi, Brézillon décrit l'évolution de ces systèmes [18] qui ne s'intéressent au départ qu'à la tâche (systèmes experts), puis à la tâche et à l'utilisateur (systèmes cognitifs conjoints), jusqu'à la tâche en situation (systèmes intelligents), avec une emphase pour le contexte dont la tâche, l'utilisateur, l'environnement immédiat etc.

Parmi ces systèmes intelligents plusieurs approches sont envisageables. Brézillon en distingue deux grands types : les systèmes sensibles au contexte et les systèmes d'assistance intelligents basés sur le contexte [17]. La définition de la conscience du contexte choisie englobe ces deux types d'approches. Ainsi nous adaptons son continuum, Fig.13 qui traite alors des systèmes conscients du contexte où l'accent est porté plutôt sur l'identification de la situation ou sur l'assistance vis à vis de la situation. Les deux nécessitent un raisonnement portant sur des éléments de contexte, qui ont tendance à être plutôt de bas niveau pour l'identification et plutôt de haut niveau pour l'assistance.

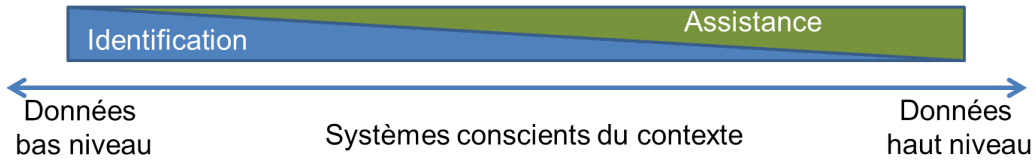


FIGURE 13: Continuum des systèmes conscient du contexte, réinteprétation de [17]

Se focaliser sur l'identification nécessite d'interpréter des données afin de classer la situation. Ce classement mène alors souvent à une assistance directe. Par exemple, un profil d'affichage est géré sur un téléphone portable dépendant du déplacement de l'utilisateur. La taille et la quantité de texte sont choisies en fonction de la situation (immobile, en voiture, marchant ou courant) qui est déterminée à partir d'un accéléromètre uniquement [95].

Se focaliser sur l'assistance permet de traiter une situation connue mais complexe, et souvent déjà décrite à l'aide de connaissances haut niveau variées. Par exemple, un conducteur de train peut à l'aide d'un système d'assistance obtenir rapidement la procédure d'arrêt d'urgence qui tient compte de l'ensemble de la situation connue (l'activité de la ligne, le lieu de la panne, le type de panne etc.) [17].

En pratique, tout comme dans le cas de la réalité mixte, tout système conscient du contexte implémente à des degrés divers une forme d'identification et une forme d'assistance (Fig.13). Nous aimerions pouvoir obtenir une bonne gestion des deux formes pour notre assistance générique à interaction 3D, afin de permettre un usage quels que soient les besoins initiaux de l'application considérée.

2.2 LES SYSTÈMES CONSCIENTS DU CONTEXTE

2.2.1 *Propriétés communes*

Les applications consciente du contexte (Fig. 16) tout en s'attardant de manière différente sur chaque élément partagent une liste commune de propriétés idéales à gérer [6] :

- L'hétérogénéité et la mobilité du contexte ;
- Les relations et dépendances entre éléments de contexte ;
- Le temps : permettre l'accès à des états passés ou futurs ;
- L'imperfection : les données peuvent être erronées voire incorrectes ;
- Le raisonnement : afin de prendre des décisions ou de dériver du contexte d'un niveau supérieur ;
- L'utilisabilité [2] des formalismes de modélisation ;
- Un accès efficace aux informations de contexte.

Un autre point de vue [69] sur ces propriétés sépare les besoins concernant la représentation et le raisonnement, bien que finalement la seule réelle différence soit la référence implicite ou explicite au besoin de gestion du temps. Les réactions au contexte peuvent être classées en trois catégories [36] :

- La présentation d'informations ou de services à l'utilisateur ;
- L'exécution automatique d'un service ;
- L'association de contexte à une entité ou une information pour un usage futur.

Néanmoins, de part la nature même du contexte qui est constamment négocié et redéfini, un design idéal ne doit pas supporter des informations particulières mais permettre son évolution [37]. Ainsi les propriétés et les réactions possibles précédentes doivent également être modifiables et extensives dynamiquement. De même au sujet des réactions, le lien entre l'action du système, le contenu, et la description haut niveau, le contexte, doit pouvoir être redéfini [3]. Enfin, le système doit pouvoir idéalement dialoguer avec l'utilisateur pour négocier le contexte. Le système doit pouvoir alors exhiber sa structure et son raisonnement afin de s'accorder ensuite avec l'utilisateur sur la notion de contexte : c'est-à-dire permettre sa modification, compréhensible voire guidée, à tout moment. D'ailleurs, même en privilégiant une vision ingénierie, la possibilité de coopération et l'échange

2. facilité d'expression et d'usage
3. Cette contrainte est essentielle pour pouvoir gérer des applications multiples notamment en effectuant peu d'hypothèses, comme pour notre système.

d'informations entre le système d'assistance et l'utilisateur sont jugées essentielles [17]. Ainsi un raisonnement totalement abstrait et l'exclusion de l'utilisateur sont des causes probables d'échec [18]. Finalement, les critères idéaux suivants sont ajoutés, en tant qu'éclairage différent ou comme propriétés supplémentaires :

- la possibilité d'accéder à la structure et aux étapes du raisonnement ;
- la possibilité de modifier dynamiquement la base de connaissance : que ce soit les informations primaires ou le raisonnement qui puissent être modifiables par l'utilisateur, l'application ou le système lui même ;
- le découplage des décisions de leur réification : permettre à chaque instance d'interpréter la décision.
- une expressivité [4] élevée pour permettre les évolutions futures.

Enfin intéressons nous au point de vue de l'adaptation. Quatre aspects sur lesquels peuvent porter l'adaptation sont séparés [84] :

- la présentation ;
- les données ou informations ;
- les services ;
- l'assistance à l'utilisateur.

Ces aspects dépendent eux-même de quatre facteurs :

- l'utilisateur (ses connaissances, ses préférences, son expérience, ses habiletés) ;
- la plate-forme d'interaction (sa puissance, ses périphériques, sa connectivité, son système d'exploitation et ses logiciels) ;
- l'environnement (les informations ambiantes, l'aspect mono/multi-utilisateurs, la structure, les politiques de sécurité) ;
- les caractéristiques de l'activité (les tâches, leur structure et les buts courants).

Ainsi les deux points de vue concordent bien. Les réactions au contexte ciblent les différents aspect de l'adaptation et inversement. Les facteurs sont les éléments récurrents de contexte pour l'adaptation. L'emphase des approches conscientes du contexte porte en générale d'abord sur l'activité, puis l'utilisateur et l'environnement et en dernier la plateforme [5].

2.2.2 *Domaines des applications conscience du contexte*

Les applications pouvant être classées comme conscientes du contexte sont très nombreuses. Il est difficile d'en parler de manière exhaustive. Nous présentons donc un tour d'horizon rapide des différents domaines (Fig. 16) où de telles applications sont utilisées ainsi que leurs principales problématiques. Puis dans les sections suivantes seront détaillés des exemples notables, où la réutilisation des frameworks présentés a été, à différents degrés, envisagée.

INFORMATIQUE UBIQUITAIRE En informatique ubiquitaire, l'objectif est d'exploiter et de faciliter l'usage de l'informatique nous environnant. Les problématiques principales sont [83] la communication entre appareils et l'architecture décentralisée, la possibilité de mesures erronées ou incompatibles, l'intégration à l'environnement existant et la limitation de la plateforme. C'est le domaine le plus prolifique pour la

4. une mesure des idées exprimables dans un langage

5. Plasticité, malléabilité et flexibilité peuvent être vues comme une conscience du contexte dont l'emphase serait au contraire les facteurs liés à la plate-forme d'interaction

conscience du contexte. Moduler le profil d'utilisation d'un téléphone portable en fonction de la situation identifiée en est un exemple [95].

AGENTS INTELLIGENTS Les agents intelligents peuvent être physiquement présents dans le monde réel (robot etc.), dans le monde virtuel (personnage autonome etc.) ou être dissimulés (agent logiciel etc.). Les problématiques principales sont la communication, la gestion d'un contexte local, la mémoire de l'agent, la crédibilité ou l'habilité à reproduire un comportement humain. Un exemple d'agent dissimulé permet dans un salon d'exposition virtuel [21] et à l'aide de machines d'états communicantes, d'effectuer des raccourcis pour l'interaction avec certains objets spécifiques, en traitant l'historique de ces interactions.

ENVIRONNEMENTS SÉMANTIQUES Un environnement sémantique stocke sa propre description, comme le web sémantique. On parle également d'environnement virtuel informé [86]. Les problématiques principales sont de rendre l'environnement compréhensible par la machine, de gérer les environnements de taille importante, de servir de base à une intelligence ou d'introduire de la sémantique au coeur de la boucle de rendu graphique [56]. Une application en est la construction automatique de l'environnement ou sa personnalisation. PeVEP (Personalized Virtual Exhibition Platform) [9] permet de pré-organiser le rendu d'une collection de musée dépendant du profil utilisateur. Une de ces fonctionnalités est d'estimer le profil utilisateur de manière automatique et donc l'application complète appartient au domaine des environnements virtuels intelligents (voir ci-dessous).

L'INTERACTION 3D ADAPTATIVE L'interaction 3D adaptative tente d'assister l'utilisateur lors de son interaction avec un environnement 3D. Une localisation spécifique d'une adaptation comme la technique de sélection Go-Go [70] (où l'interaction dépend de la position de la main) en est un bon exemple. Les formes d'interaction 3D adaptatives sont variées : en font partie dans une certaine mesure la reconnaissance de gestes [34], qui associe un même sens à différents mouvements, ou le processus levant l'ambiguïté d'entrées utilisateur [49], qui associe un sens différent à une même commande selon le contexte.

ENVIRONNEMENTS VIRTUELS INTELLIGENTS Les environnement virtuels intelligents sont définis comme un mélange d'intelligence artificielle, de vie artificielle et de réalité virtuelle [1]. Ce concept se superpose à une grande partie des domaines d'applications à l'exception de l'informatique ubiquitaire. Les problématiques sont donc les mêmes avec un accent particulier sur les capacités de raisonnement. Cela inclut les applications où une intelligence gère une information sémantique riche issue de l'environnement. Par exemple les systèmes d'apprentissages intelligents [41], la génération d'environnement [9] ou un environnement sémantique incluant son raisonnement [56]. Dans tous ces cas, l'historique du contexte est d'un grand intérêt.

INTELLIGENCE AMBIANTE L'intelligence ambiante [71] est également un domaine multidisciplinaire basé sur l'informatique ubiquitaire, l'intelligence artificielle et les objets sémantiques. C'est à peu près la version dans le monde réel des environnements virtuels intelligents. Les deux domaines ne sont pas totalement séparés puisque l'on peut utiliser des capteurs physiques intelligents pour la réalité virtuelle ou afficher un environnement intelligent sur un smart-phone. Par exemple, le projet

GAIA [72] propose un framework avec de nombreuses méthodes de raisonnement pour créer une maison intelligente.

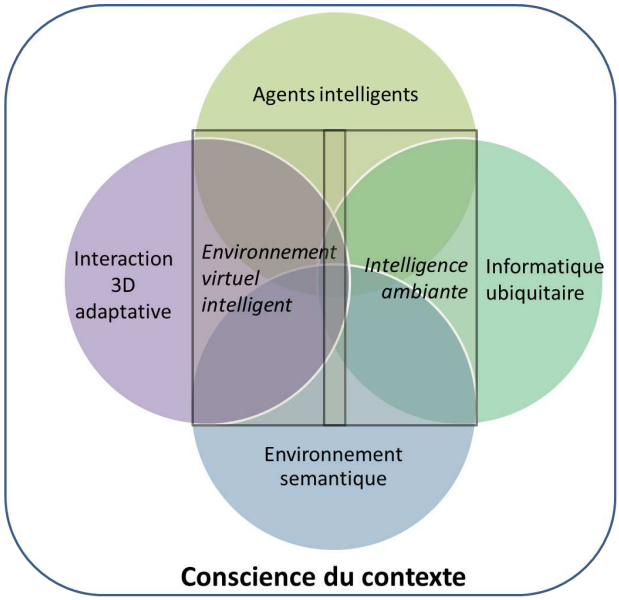


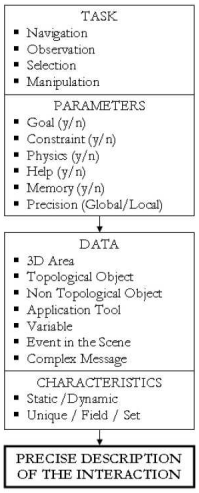
FIGURE 14: Différentes familles d'applications consciente du contexte

Ainsi de très nombreux travaux utilisent une notion plus ou moins élaborée de contexte. Des descriptions de travaux supplémentaires peuvent être trouvées dans différents états de l'art [16] [25][45][83][2][7][69][6]. Ils sont en général orientés informatique ubiquitaire et/ou intelligence ambiante qui sont les domaines les plus prolifiques pour la conscience du contexte. De nombreux travaux pourraient être réutilisés et répondent partiellement à notre problématique. Une sélection de ces travaux est présentée par la suite.

2.3 EXEMPLES DE CONSCIENCE DU CONTEXTE POUR L'INTERACTION 3D

2.3.1 *La supervision du rendu multimodale selon la tâche*

Bouyer et al. propose une approche générique et contextuelle du rendu multimodal en réalité virtuelle à l'aide d'un superviseur. Pour cela, le rendu le plus adéquat est déterminé selon la sémantique des tâches, les caractéristiques des canaux et des modalités ainsi que le contexte de l'interaction en cours. La description du contexte de l'interaction est effectuée à travers une liste de descripteurs (Fig 15a) pour la tâche (e.g. navigation), les paramètres (e.g. précision), le type de donnée (e.g. zone 3D) et des caractéristiques (e.g. statique). L'assistance obtenue est le choix d'un ou plusieurs rendus adaptés, combinant le canal sensoriel utilisé ainsi que la modalité et ses attributs (Fig. 15b). Cette description est utilisée par un moteur de logique du premier ordre, Prolog, qui compare un ensemble de scores avant de choisir le ou les meilleurs rendus, selon le niveau utilisateur (Fig. 16a). Le moteur est externe et communique avec la scène virtuelle (Fig. 16b).

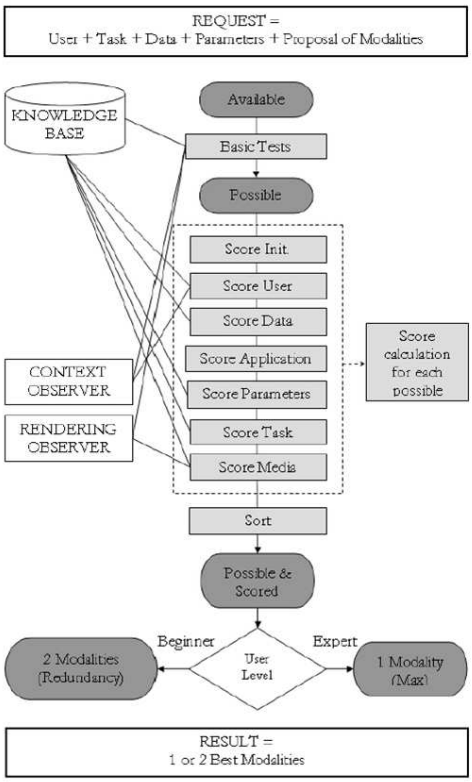


(a) Représentation de l'interaction

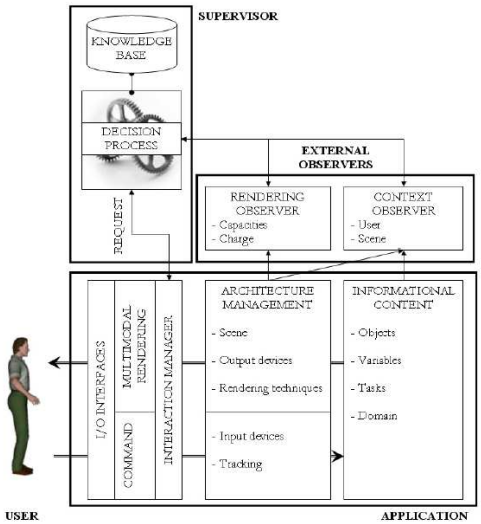
Channel	Modalities	Attributes
Visual	Topology, Physics, Text, Colour, Mathematics, Widgets, Levels of Detail, Camera, Guide	Location, Time, Intensity, Size
Audio	Earcon, Music, Auditory Icon, Vocal Message,	Location, Time, Frequency, Volume, Timbre
Haptic	Guide, Physics, Spring, Texture, Vibration	Time, Intensity, Direction

(b) Représentation des assistances

FIGURE 15: Représentations pour le superviseur multimodal [12]



(a) Raisonnement



(b) Architecture

FIGURE 16: Raisonnement et architecture du superviseur multimodal [12]

Ce moteur cible ainsi directement l'assistance à l'interaction 3D et la rend adaptative en se basant sur un contexte. C'est un processus externe qui communique avec l'application. Le raisonnement à l'aide de logique du premier ordre permet une bonne expressivité. En revanche le processus est très spécialisé et ne se préoccupe pas d'identifier le contexte. La requête doit ainsi inclure la situation détaillée à l'aide des descripteurs initiaux. De plus la représentation choisie (des prédicats logique, utilisé comme des paires attribut/valeur évoluées) limite l'utilisabilité. Des

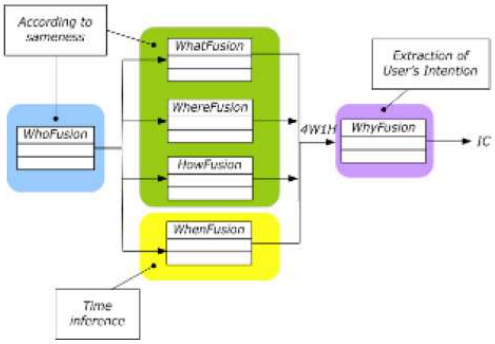
extensions via de nouvelles formes de contexte ou afin d'étendre le raisonnement, pour obtenir également une identification de la situation, sont difficiles à mettre en oeuvre bien que possible. L'approche externe, ciblant l'assistance à l'interaction et permettant une bonne expressivité grâce à la logique du premier ordre sont des éléments souhaitables pour notre propre approche.

2.3.2 *VR-UCAM : un framework pour la conscience du contexte en la réalité virtuelle compatible avec l'informatique ubiquitaire*

Un modèle de gestion du contexte VR-UCAM [55] a été pensé pour tenir compte du contexte réel ou virtuel en étendant un précédent framework pour l'informatique ubiquitaire, Ubi-UCAM [50]. Il a ainsi la particularité de gérer le contexte de manière uniforme pour la réalité mixte. Il possède notamment une architecture initialement distribuée apte à récolter des informations pour le domaine ubiquitaire (Fig 18a). La représentation est basée sur des n-uplets répartis selon le modèle W5H (Fig. 17a) et a pour vocation de représenter une large gamme de contexte. La description initiale peut être complétée à partir de l'identifiant initial par une étape de fusion de l'information, permettant par exemple de déterminer le *Pouquoi* à partir des autres descripteurs (Fig. 17b). Le raisonnement final consiste en un appariement entre la situation identifiée et un ensemble de réaction adaptée (Fig 18b).

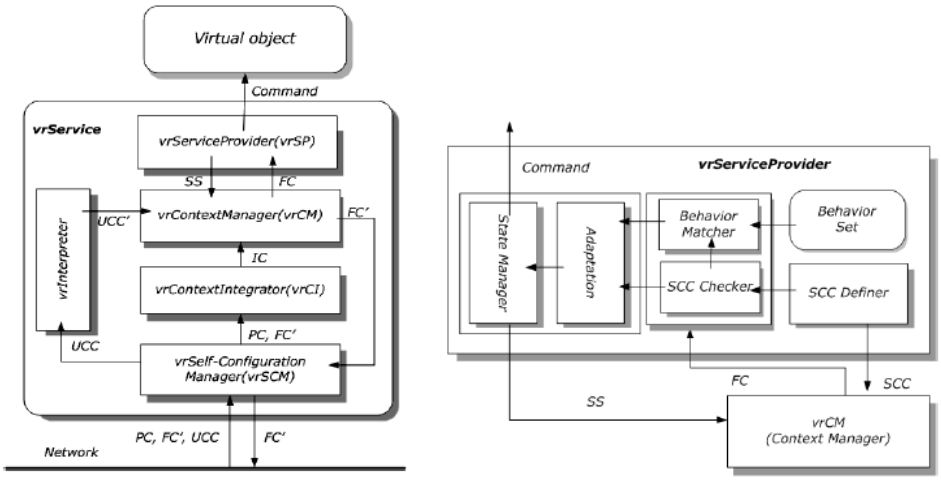
Context	Field	Definition
Who	String User_Name Double User_ID_Num	User name User ID
What	String object_Name Double object_ID	Object name Object ID
Where	URL Place_URL Coordinates (X,Y,Z)	Location URL Position
When	Enumeration Time Time Absolute_Time	Relative time Absolute time
How	int Action_ID	Behavior ID
Why	Enumeration emotion	Emotion

(a) Représentation élémentaire [55]



(b) Représentation combinée [64]

FIGURE 17: Représentation dans VR-UCAM



(a) Architecture (b) Fournisseur de service final

FIGURE 18: Architecture et raisonnement dans VR-UCAM [64]

Permettant une continuité du contexte pour les services ubiquitaires et ceux propres à la réalité virtuelle, VR-UCAM a une représentation à vocation générale. Ainsi le contexte est réparti suivant des concepts permettant a priori de décrire simplement toute situation. La description de ces concepts est cependant en pratique à nouveau des n-uplets et n'est donc pas facile à réutiliser et à étendre. Contrairement au superviseur multimodal, ce sont les assistances qui sont ici moins développées au profit d'une identification de la situation à partir d'un contexte plus général. Le modèle n'offre alors pas les mêmes capacités de raisonnement pour gérer des assistances variées. L'architecture distribuée est un plus qui n'est pas notre priorité. L'approche reste intéressante pour notre application notamment par l'introduction d'une notion de fusion intermédiaire. Elle propose ainsi de remonter à une intention simplifiée grâce à cette étape. De plus elle dénote d'une volonté de description du contexte plus large, via un panel de concepts permettant d'organiser les attributs gérés.

2.3.3 *VR-DeMo : Conception flexible et personnalisation de l'interaction 3D*

Octavia et al. s'appuient sur des diagrammes d'états, MBUID (model-based user interface design) intégrés à VR-DeMo, afin de décrire de manière flexible et dépendant du contexte un comportement dans un environnement virtuel. Ces diagrammes implémentent un raisonnement de type événement-condition-action. Le changement, la détection de changement et l'état du contexte sont gérés. Ils permettent au designer un prototypage rapide et haut niveau d'environnements conscients du contexte. Les diagrammes sont déclenchés par des évènements qui orientent les processus listés en fonction du contexte. Par exemple (Fig. 19) un paramètre, la viscosité du retour de force, est modifié en fonction du tremblement de la main de l'utilisateur. D'ailleurs, le moteur inclut un modèle utilisateur standard et un modèle personnalisable à l'aide de la base de connaissance avec historique. L'adaptation est composée de plusieurs parties intelligentes permettant de sélectionner les techniques adaptées au contexte (mapping), d'activer ou de désactiver des techniques selon l'usage et l'objet (predicting) et également d'ajuster

une technique (adaptating), Fig. 20. Les effets de l'adaptation automatique[6] sur l'utilisateur ont également été évalués. Des effets positifs malgré les changements de techniques automatiques sont notées [63].

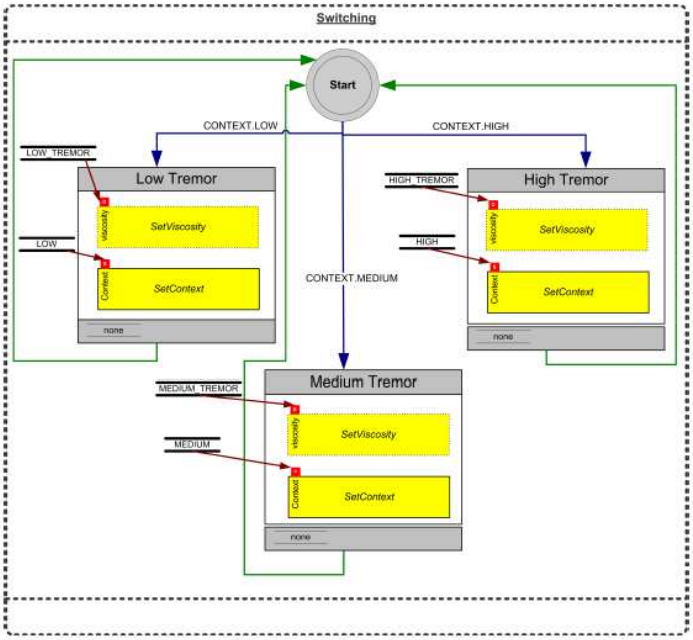


FIGURE 19: Représentation et raisonnement à l'aide de MBUID [62]

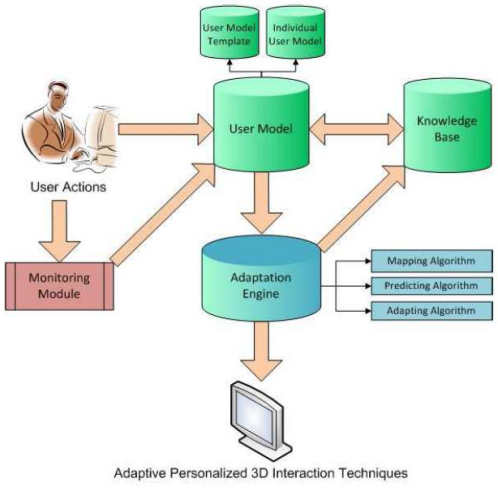


FIGURE 20: Architecture des interactions adaptatives obtenues avec MBUID [62]

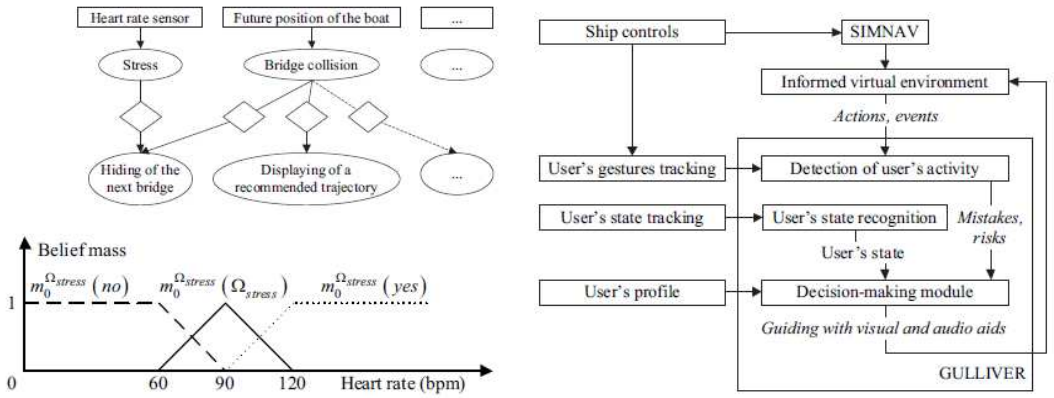
Cette approche permet la création rapide d'application consciente du contexte grâce à une gestion haut niveau. Le modèle gère l'évolution d'un processus complet intégrant le contexte. De plus, la gestion de la personnalisation, bien que souvent présente, est ici pleinement intégrée. Contrairement aux approches précédentes, l'utilisabilité est ici très grande, tant pour le raisonnement que pour la représentation. Néanmoins, c'est une approche générique modifiable de la conception, et non pas de la gestion, d'interaction 3D adaptative. Ainsi la représentation ou le

6. L'acceptation et les effets sur l'utilisateur d'adaptations automatiques non limités à l'interaction 3D sont détaillées dans de très nombreux cas [29]

raisonnement ne peuvent pas être modifiés de manière dynamique, et sont intégrés à l'application à l'issue de la conception. Il y a de nombreux points communs entre ces travaux et notre approche désirée notamment la volonté d'englober les possibilités d'interaction 3D adaptative. L'approche est intéressante pour son intégration de la personnalisation ainsi que par la grande utilisabilité que procure l'expression par graphes.

2.3.4 *GULLIVER : GUiding by visuaLization metaphors for fluviaL navigation training in Informed Virtual EnviRonment*

Le système de prise de décision GULLIVER permet la personnalisation d'un environnement de formation virtuel et est utilisé actuellement pour l'apprentissage de la navigation fluviale. Il permet notamment la gestion dynamique d'assistances et de difficultés selon l'état de stress de l'utilisateur et la prévision de son comportement. Les assistances adaptatives sont obtenues par la description des situations et des réactions attendues à l'aide de réseaux[7] de fonctions de croyance (Fig. 21a). L'architecture du système est représentée Fig. 21b. Une filtrage final permet de respecter les contraintes comme la compatibilité mutuelle des décisions ayant une forte croyance ou afin d'éviter la surcharge de l'affichage.



(a) Représentation et raisonnement (b) Architecture

FIGURE 21: Représentation, raisonnement et architecture pour GULLIVER [41]

Les avantages de cette approche sont multiples. Le raisonnement se fait en représentant des règles sous une forme graphique gérant un transfert de croyance. Le raisonnement permet la gestion des cas d'ignorance et de l'incertitude plus généralement. Bien que spécifique à l'assistance au pilotage dans cette implémentation, la méthode peut être réemployée facilement pour d'autres applications. En revanche les approches par réseaux ne permettent pas aisément la modification dynamique ou incrémentale. Il faut étendre le réseau existant pour étendre le raisonnement. L'approche est intéressante pour sa gestion de la personnalisation et des incertitudes notamment la quantification de pertinences différentes des règles. De plus est exprimée la volonté de rester paramétrable aisément grâce à une représentation graphique et l'expression d'a priori plus intuitifs que pour les réseaux bayésiens. Enfin les problématiques des compatibilités et des surcharges dues aux adaptations sont également abordées.

7. Ils sont à la théorie des possibilités ce que sont les réseaux bayésiens aux probabilités.

2.4 EXEMPLES DE CONSCIENCE DU CONTEXTE POUR L'INFORMATIQUE UBIQUITAIRE

2.4.1 *GAIA : Une infrastructure pour la conscience du contexte basée sur la logique du premier ordre*

GAIA [72] est une approche générique qui vise à obtenir une conscience du contexte, notamment pour des applications de l'intelligence ambiante. Le contexte est décrit à l'aide de prédicats (que l'on peut voir comme des n-uplets, Fig. 22a) et le raisonnement est effectué en logique du premier ordre (Fig. 22b). Elle permet un raisonnement complexe sur les informations de contexte et offre une mécanisme pour qu'une application ou l'utilisateur puisse facilement définir ses comportements en ajoutant ses propres règles, afin d'étendre ou de remplacer le comportement initial. Le système possède une architecture distribuée qui permet de réunir les informations de différents capteurs et de les répartir aux applications directement ou après traitement (Fig. 23).

Location (Manuel, In, Room 2401)
Location (Chris, In, Room 2401)
Location (Roy, Entering, Room 2401)
Temperature (Room 2401, =, 68)
Lighting (Room 2401, Is, Dim)
Application (PowerPoint, Is, Running)
Application (MP3 Player, Is, Running)

(a) Représentation

#People(Room 2401, ">=", 1) **AND** Application (MPEG Player, Running) => RoomActivity(2401, Movie Screening)
#People(Room 2401, ">=", 3) **AND NOT** $\exists_{\text{Application}}$ x Application(x, Running) => RoomActivity(2401, Meeting)

(b) Règle de raisonnement

FIGURE 22: Représentation et raisonnement pour GAIA [72]

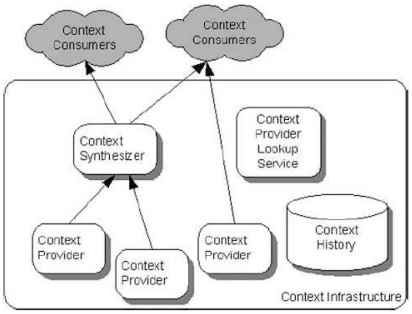


FIGURE 23: Architecture pour GAIA [72]

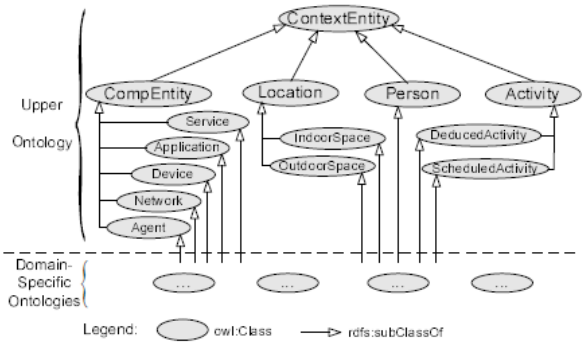
Cette approche possède un raisonnement expressif à l'aide de la logique du premier ordre. L'approche initiale est étendue par la suite à l'aide de planning et d'autres possibilités de raisonnement comme les réseaux baysésiens [6]. Cette approche est intéressante pour de nombreuses raisons. Premièrement elle propose ainsi en pratique tant l'identification que l'assistance vis-à-vis du contexte. De plus, elle prévoit la gestion dynamique des informations mais également des règles. Ce mécanisme permet de se rapprocher de la négociation dynamique intrinsèque au contexte. De plus, l'appel d'autres possibilités de raisonnement permet également de cumuler les avantages de différentes approches, notamment la gestion de l'incertain par les réseaux bayésiens. C'est enfin une approche gérant le planning et possédant ainsi une gestion du temps plus évoluée (se rapprochant des propriétés idéales de la conscience du contexte). En revanche, un points négatif est que la modification et l'expansion d'une représentation basée sur des n-uplets ne sont pas aisées.

2.4.2 *SOCAM : Un middleware pour des services conscients du contexte*

SOCAM (Service-Oriented Context-Aware Middleware) vise à obtenir une intelligence ambiante par une gestion du contexte [45]. Une description ontologique, à l'aide du langage OWL (Fig. 24c), est utilisée et permet ainsi de classer les concepts utilisés (Fig. 24b). Ces concepts peuvent ensuite être utilisés pour raisonner. Cela permet d'obtenir une représentation réutilisable et extensive. Le raisonnement classique à l'ontologie, la logique de description, est spécifique et permet de classer une situation selon un concept englobant compatible. Elle limite en revanche les possibilités de raisonnement par rapport à d'autres logiques. Cependant elle permet de facilement vérifier la compatibilité d'une information avec la base de connaissance. La logique du premier ordre est utilisée ici en supplément pour le raisonnement (Fig. 24c). L'approche est distribuée pour répondre aux besoins de l'intelligence ambiante (Fig. 25). A noter que l'approche s'attarde à inclure de nombreuses expressions de formes d'imprécision à l'aide de l'ontologie afin de répondre aux multiples besoins rencontrés lors de l'usage de capteurs hétéroclites.

```
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasChildren"/>
      <owl:to Class rdf:resource="#Person"/>
      <owl:classifiedAs rdfs:resource="http://lucan.ddns.comp.nus.edu.sg/octopus/classification#Defined"/>
    </owl:Resrtiction>
  </rdfs:subClassOf>
</owl:Class>
```

(a) Description du concept de personne en OWL



(b) Classification effective des concepts obtenue par l'ontologie

$$Location(John, MasterBedRoom) \wedge Posture(John, LiedDown) \wedge Status(Door, Close) \wedge (\neg Status(Curtain, Open)) \vdash Status(John, Sleeping)$$

(c) Règle en logique du premier ordre

FIGURE 24: Représentations et raisonnement dans SOCAM [45]

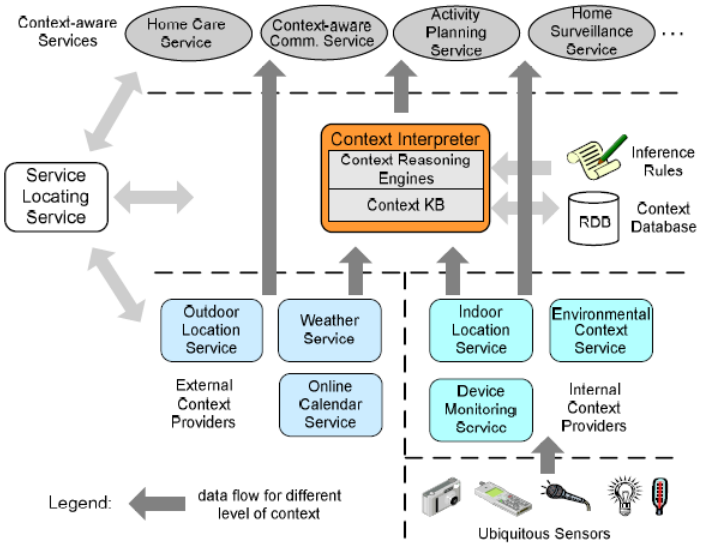


FIGURE 25: Architecture dans SOCAM [45]

L'approche a des similitudes avec GAIA pour l'usage de la logique du premier ordre, l'architecture distribuée et les applications visées. Mais une des différences majeures est qu'elle utilise une ontologie. Cela est intéressant afin d'obtenir une gestion réutilisable et extensive du contexte. L'expression peut se faire via des concepts, ce qui la rend plus aisée. De même un nouveau concept peut être classé dans la hiérarchie existante et ainsi bénéficier du raisonnement concernant les concepts parents. Cela permet de répondre à certaines de nos problématique, y compris de pouvoir considérer in fine une communication avec l'utilisateur via les termes de cette ontologie. Un point négatif est la nécessité de faire co-exister deux représentations différentes pour les formes possibles d'informations.

2.4.3 *MoBe : une architecture générale pour la conscience du contexte distribuée*

Mobile Being (Mobe) est une architecture pour l'informatique ubiquitaire. Elle combine une approche ontologique et le traitement de valeurs concrètes par des règles afin d'établir une première série de concepts abstraits (également décrits dans l'ontologie). Les concepts établis sont alors traités par un réseau bayésien permettant de déduire la situation courante (Fig.26). L'architecture est orientée pour l'informatique ubiquitaire avec notamment une gestion de la sécurisation des données. L'architecture est générale (Fig.27) et peut être connectée à différents frameworks permettant la réalisation d'applications diverses (les MoBeLets).

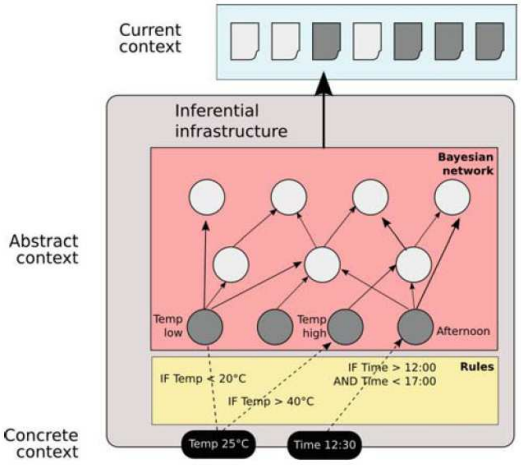


FIGURE 26: Représentation et raisonnement dans MoBe [28]

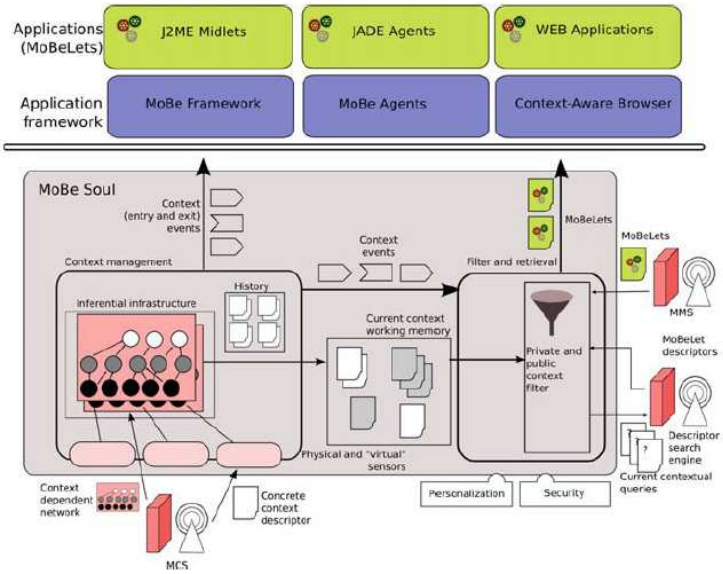


FIGURE 27: Architecture de MoBe [28]

Cette architecture combine de nombreuses approches et avantages. Notamment la gestion de raisonnement incertain avec les réseaux bayésiens, une approche lisible et extensible via l'ontologie qui est connectée au monde réel à travers des règles. Cependant, Coppola et al. précise que la taille du réseau bayésien est dépendant du nombre de variables et donc de capteurs [8]. Un réseau complètement général devrait ainsi inclure l'ensemble des variables et des concepts possibles. Une solution proposée est de le diviser suivant une dimension, par exemple la localisation, et d'acquérir le réseau adapté à chaque emplacement (dans le cadre de l'informatique ubiquitaire). MoBe est intéressant de par la multiplicité des approches mises en oeuvre. Cependant la gestion de graphe complet pour chaque possibilité suivant une ou plusieurs dimensions n'est pas forcément aisée. En revanche l'approche prévoit ainsi une forme de modification du raisonnement dynamique. Cette modification n'est cependant pas incrémentale et le classifieur de situations est en réalité changé dans son ensemble. Cette approche se focalise finalement sur l'identification de la

8. Un des inconvénients de l'approche déjà rencontré [94] dans l'introduction aux adaptations en interaction 3D.

situation plus que sur les assistances possibles. Néanmoins elle possède un attrait particulier, en tant que moyen général d'extraire des informations sémantiques d'un cas concret, grâce à la combinaison de règles, d'ontologie et d'un raisonnement creux à entrainer selon la situation, les réseaux bayésiens. Cette classification peut ensuite être utilisée dans une étape de gestion d'assistance.

2.5 LES ENVIRONNEMENTS SÉMANTIQUES COMPLETS

Cette approche est à part bien qu'elle touche à l'interaction 3D. En effet, elle va au delà de l'obtention d'une conscience du contexte et modifie la compréhension et l'autonomie des environnements virtuels. Pour cela, il est possible de modifier le fondement même de leur représentation et d'inclure une description haut niveau de l'environnement, à l'aide de graphes sémantiques [56]. Le lien entre cette représentation et la représentation graphique gérée par le moteur de rendu doit alors être alors ajouté (Fig.28). Le processus complet utilise trois moteurs : un moteur d'inférence, un moteur de comportement et un moteur de rendu graphique (Fig.29). L'interprétation de la situation est effectuée par le moteur d'inférence en s'appuyant sur la description sémantique et permet d'obtenir des raisonnements de sens commun (e.g. la destruction d'un objet qui en supportait un autre, met à jour la représentation sémantique et détache des graphes précédemment liés). Le comportement à effectuer dépend de conditions potentiellement présentes dans les graphes décrivant la situation (e.g. un objet non supporté par un autre objet tombe). Le rendu et les évènements physiques sont gérés par le moteur graphique de la scène. Un exemple de fonctionnement du processus complet est présenté Fig. 30. Les moteurs d'inférence et de comportement utilisent le raisonnement qualitatif, qui s'attache à décrire les aspects continus du monde à travers des informations qualitatives (e.g. élevé) plutôt que quantitative, à l'aide de règles.

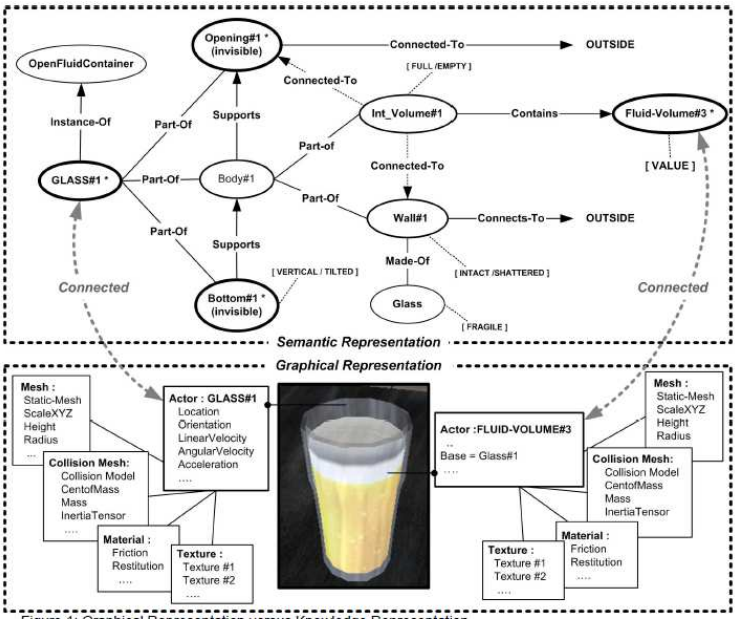


FIGURE 28: Lien entre les représentations graphique et sémantique d'un monde virtuel [56]

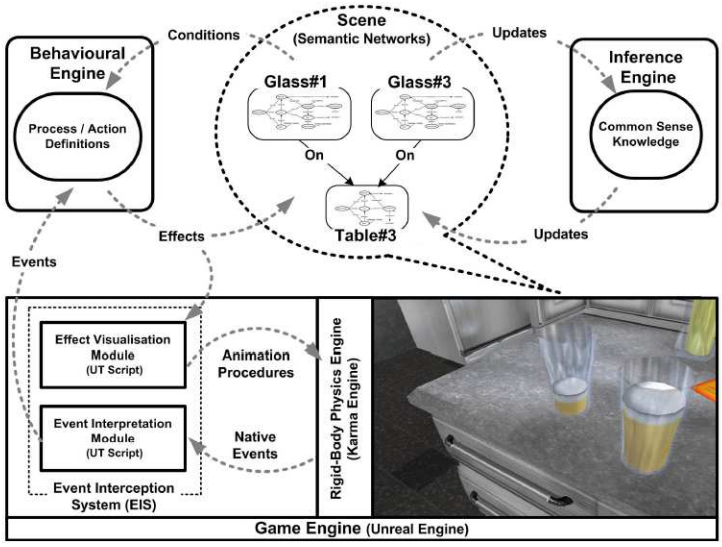


FIGURE 29: Architecture pour la gestion de l'environnement sémantique virtuel [56]

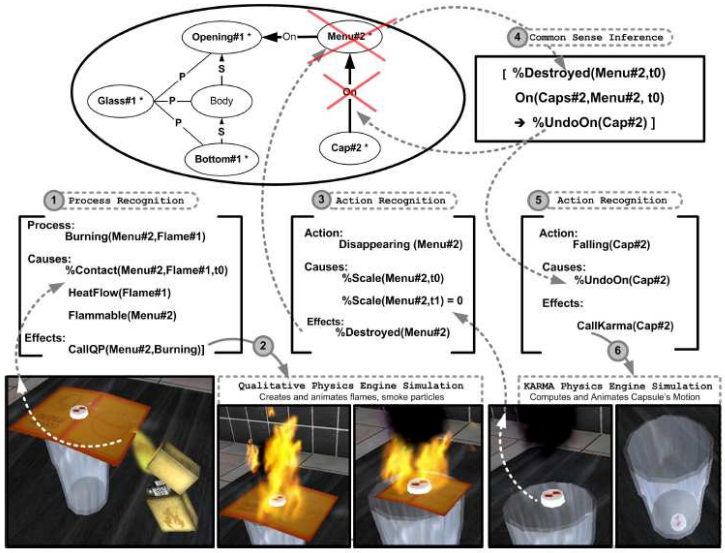


FIGURE 30: Exemple de raisonnement de l'environnent sémantique virtuel [56]

L'inclusion de sémantique est un domaine de recherche en soi où une des difficultés majeures est la définition et l'inclusion de différentes granularités de descriptions. Par exemple, Gutierrez et al. [46] propose un modèle d'environnement sémantique. La description sémantique ne porte pas uniquement sur la description d'objets 3D et de leurs relations topologiques mais comprend également une description sémantique de leurs capacités d'interactions, Fig 31. Chaque item virtuel informe l'environnement de ses compétences et fonctionnalités et accepte les interactions de manière transparente. Ces items jouent à la fois le rôle de contenu et d'interface pour l'environnement virtuel. Par exemple l'animation et le comportement à effectuer, de manière autonome ou selon le stimuli, sont intégrés au modèle.

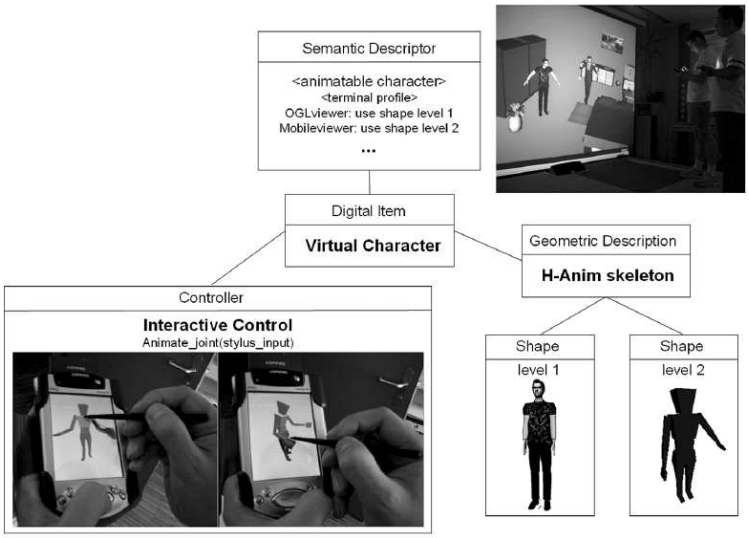


FIGURE 31: Modèle de description d'environnement sémantique [46]

Ce domaine de recherche est très prometteur et est un nouveau paradigme de représentation et de raisonnement. Cependant en se concentrant sur l'interaction 3D adaptative, nous aimerions pouvoir obtenir des adaptations automatiques dans la majorité des applications, qui n'intègrent pas (pour l'instant) de telles descriptions sémantiques. Ainsi nous partons du principe que le raisonnement à obtenir pour gérer et comprendre le contexte doit extraire une sémantique de l'environnement sans en supposer la présence au préalable, comme le fait Mobe [28]. Cependant l'approche via une représentation à l'aide de réseaux sémantiques et un raisonnement qualificatif est intéressante tant pour l'expressivité que pour l'utilisabilité.

2.6 POSITIONNEMENT DE NOTRE APPROCHE

Notre problématique principale est l'obtention générique d'interactions 3D adaptatives. Et pour atteindre une adaptativité riche tout en limitant nos a priori sur l'application, une gestion idéalement complète du contexte est nécessaire. De nombreuses propriétés sont ainsi à gérer (section 2.2.1). En résumé, le système doit offrir expressivité, extensibilité, utilisabilité, modification et évolution dynamique des connaissances et du raisonnement.

De nombreuses approches dans des domaines différents abordent implicitement ou explicitement la conscience du contexte. Mais aucune approche n'implémente l'ensemble des propriétés idéales. Il est d'ailleurs difficile de comparer l'ensemble des approches rencontrées, dont certaines sont ici brièvement résumées. Chacune propose en effet une réponse à une problématique qui n'est pas entièrement la notre et répond à des degrés variables à différents critères idéaux. Parfois des contraintes supplémentaires sont ajoutées par rapport à notre objectif, comme une gestion distribuée qui est importante pour l'informatique ubiquitaire. Mais certaines contraintes sont également allégées comme le potentiel d'extensibilité ou l'accès et la modification dynamique de la représentation et/ou du raisonnement. D'autres enfin, comme la gestion d'environnement sémantique complet, potentiellement le futur paradigme des environnements virtuels, ont simultanément une contrainte et un avantage supplémentaire de taille : cette même description sémantique complète

de l'environnement.

Il est ainsi envisageable de se baser sur des approches présentées. Mais chacune a des avantages et des inconvénients par rapport à notre objectif (Fig. 32).

Approche	Avantages	Inconvénients	Intérêts particuliers
Superviseur multimodal [12]	Gestion multimodale générique & expressivité.	Pas d'identification de la situation & peu facile à étendre.	L'assistance externe à l'interaction 3D & l'expressivité
VR-UCAM [55]	Approche générale pour la réalité mixte.	Peu facile à étendre & raisonnement peu expressif	L'idée de fusion intermédiaire & la volonté de description générale.
VR-DeMo [62]	Prototypage rapide générique.	Difficultés de modifications dynamiques.	La grande utilisabilité à l'aide de graphes & la personnalisation.
GULLIVER [41]	Gestion modifiable & paramétrable d'assistances	Difficultés de modifications dynamiques & d'extension.	L'usage de graphes intuitifs, l'incertain & la personnalisation.
GAIA [72]	Expressivité & modification dynamique.	Peu facile à étendre.	Les raisonnements multiples & la modification dynamique.
SOCAM [45]	Approche sémantique & expressivité.	Représentations différentes nécessaires.	L'approche sémantique via l'ontologie. Raisonnement renforcé par la logique du premier ordre.
MoBe [28]	Classifieur générique & utilisable	Extension & modification du raisonnement	L'approche générale pour classifier une situation.
Environnement sémantique [56]	Réprésentation & expressivité poussées	Sémantique obligatoire dans l'environnement	L'approche sémantique par graphes & le raisonnement qualitatif

FIGURE 32: Comparatif informel des méthodes présentées

Ainsi une combinaison des approches, en sélectionnant des intérêts particuliers de chacune, est envisagée. De plus, en pratique une approche ascendante est toujours nécessaire à un moment ou à un autre pour la conception de systèmes d'assistances, en effet *on ne sait pas ce qui est réellement nécessaire jusqu'à ce que l'on entame le processus de création* [48]. Nous allons ainsi nous intéresser à la représentation et au raisonnement pour les systèmes conscients du contexte. Nous allons pouvoir comparer qualitativement sous cet angle les approches rencontrées et choisir la notre. Cela nous permettra de proposer alors une assistance générique consciente du contexte pour l'interaction 3D qui soit :

- expressive : capable de représenter et de raisonner sur un contexte quelconque ;
- utilisable, extensive et modifiable dynamiquement tant pour la représentation que pour le raisonnement ;
- intégrable par la majorité des applications et fonction de leur possibilités ;.

3

MODÉLISATION ET FORMALISATION POUR LA CONSCIENCE DU CONTEXTE

Résumé : *La construction d'un raisonnement sémantique propre pour répondre à notre cahier des charges est envisagé. Pour cela, les différentes possibilités de modélisation sont considérées. Premièrement, nous nous intéressons aux types de modèles rencontrés pour la conscience du contexte. Puis à ceux de l'adaptation qui détaillent des formes pratiques d'assistances issues du contexte. Les différents formalismes d'implémentations sont alors considérés. Une représentation de bas niveau favorise l'efficacité mais est difficile à lire et à modifier. L'ontologie facilite la compréhension et la réutilisabilité mais n'offrent pas en soi de très bonnes possibilités de raisonnement. Les graphes conceptuels, basés sur une ontologie, permettent de représenter l'ensemble des situations et de raisonner sur celles-ci. Ils sont utilisables, faciles à comprendre et à modifier ainsi qu'expressifs. La plateforme Amine, offrant notamment une couche de logique du premier ordre intégrant les graphes conceptuels, est alors considérée pour notre implémentation.*

3.1 MODÉLISATION DES APPLICATIONS CONSCIENTES DU CONTEXTE

Afin d'obtenir une assistance automatique à l'interaction 3D, il y a deux objectifs imbriqués à réaliser. Il faut accéder au contexte et réagir en fonction de celui-ci. Quelle que soit la modélisation du système choisie, elle influence les possibilités de définitions des éléments constituant le contexte, leur obtention et leur représentation ainsi que le raisonnement applicable et les réactions décidables. Différents aspects de la modélisation sont donc étudiés.

3.1.1 *Lien avec l'intelligence artificielle*

Les différents domaines de l'intelligence artificielle correspondent à des capacités dont doit se doter un agent intelligent afin de penser et/ou agir, de manière rationnelle et/ou humaine. L'intelligence artificielle se divise en sept grands domaines [3] :

- la représentation des connaissances : structure de données, représentation de l'incertitude, extraction de données pertinentes ;
- la planification et la résolution de problèmes : si A est vrai et A implique B, alors B est vrai ; que faire pour prouver B ;
- l'inférence : générer une hypothèse à partir de données partielles ;
- la compréhension et l'interprétation du langage naturel ;
- l'apprentissage : à partir d'exemples, de punitions, par imitation etc ;
- les méthodes de recherche (ex : échecs) ;

– la vision, le traitement d'image et la reconnaissance de forme [1]

L'ensemble des sous-domaines peuvent être utiles pour la réalisation du moteur d'assistance, mais fondamentalement sont nécessaires la représentation des connaissances ainsi que la résolution des problèmes. Nous parlons globalement de raisonnement pour les domaines autres que la représentation des connaissances. En pratique les deux ne sont pas totalement indépendants. Par exemple, profiter de la logique floue nécessite d'avoir inclus une fonction d'appartenance dans la représentation. Ainsi le choix de la représentation a un impact sur les possibilités de raisonnement. De plus une représentation expressive mène en général à des problèmes plus complexes et les algorithmes de raisonnement sont alors moins efficaces. L'expressivité est utilisée comme une mesure des idées exprimables dans un formalisme, indépendamment de la facilité d'expression. L'utilisabilité [2] reflète l'aisance d'expression et plus généralement d'usage.

3.1.2 *Catégories de modèles en conscience du contexte*

Les différents formalismes pour obtenir la conscience du contexte peuvent être classées, avec ainsi [45][83][2] [69] :

– Les modèles à attribut/valeur et les modèles à balises. De structures simples et orientés applications, ils sont souvent utilisés pour décrire les capacités d'un service, sélectionné par appariement. Les balises permettent de hiérarchiser à l'aide d'étiquettes la structure. Ce sont en pratique le type d'entrées de contexte pour VR-UCAM [55] et le superviseur multimodal et [12] bien qu'englobées respectivement dans les concepts du W5H et des prédicats.
– Les modèles graphiques. Ces modèles permettent notamment le prototypage rapide d'applications et offre une plus grande utilisabilité pour le designer, par exemple MBUID dans VR-DEMO [62] ou les réseaux évidentiels dans GULLIVER [41].
– Les modèles orientés objets. Les modèles encapsulent les détails de la gestion locale de contexte et permettent son accès via des interfaces spécifiques. Par exemple, VR-UCAM [55] qui réparti des objets contextualisés communicant avec l'ensemble de l'architecture distribuée.
– Les modèles logiques. Ils ont un degré très élevé de formalisme et d'expressivité. Des faits, des expressions et des règles définissent le contexte. Une des premières approches [57] formalise au passage des propriétés du contexte. C'est également l'approche de GAIA [72], du superviseur multimodal [12] et celle derrière les graphes de GULLIVER [41] ou ajoutée par SOCAM [45].
– Les modèles ontologiques. Ils permettent de décrire des concepts et des relations à l'aide d'une ontologie comme base de représentation et sont ainsi

1. Ce domaine peut être également perçu comme indépendant et partageant de nombreuses problématiques avec l'intelligence artificielle. Autrement l'ajout de capacités de perceptions et/ou d'actions peut être considéré comme appartenant également globalement à l'intelligence artificielle. Ainsi une autre distinction [75] sépare quatre domaines d'intérêts primaires (la représentation des connaissances, le raisonnement automatique, l'apprentissage, le langage naturel) et deux autres secondaires (la vision par ordinateur et la robotique) selon les capacités demandées à l'agent intelligent

2. L'expressivité présentée est ainsi une expressivité théorique. L'utilisabilité peut être vue comme une expressivité pratique. Il est alors discutable que l'expressivité globalement atteinte, en tant que la capacité à exprimer des idées soit conditionnée par l'expressivité théorique mais également pratique. En effet, à expressivité théorique égale, les idées réellement exprimées par un utilisateur exploiteront mieux les possibilités offerte lorsque l'utilisabilité ne freine pas sa compréhension et son expression naturelle.

extensifs, utilisables et interopérables. L'expressivité du raisonnement associé est également généralement élevée. C'est le cas en partie de SOCAM [45] et de MoBe [28]

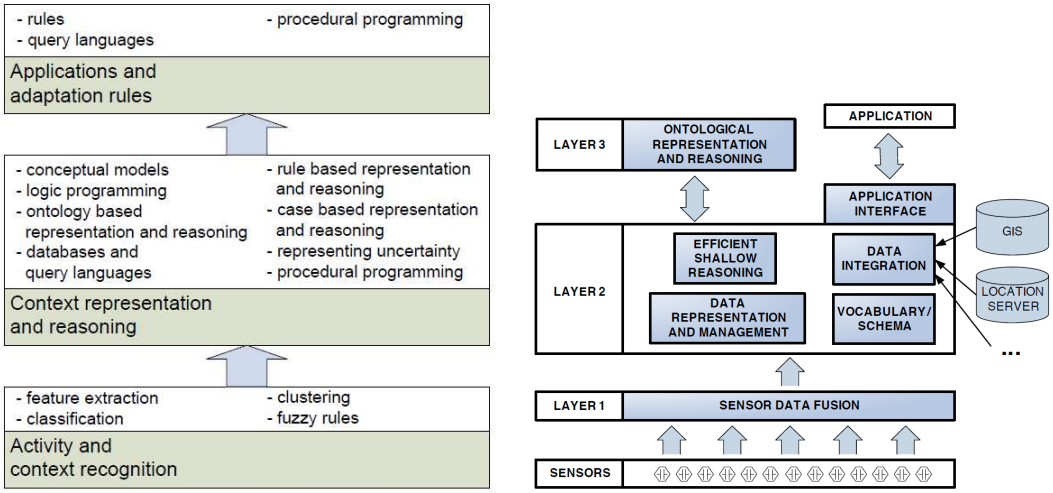
Les approches ontologiques sont les plus fréquentes actuellement [69]. De nombreuses méthodes sont mixtes et mélangent les modèles ainsi que les capacités de raisonnement et/ou de représentation. Ainsi les applications ontologiques utilisent souvent la logique de description comme capacité première de raisonnement mais sont également souvent associées à une partie supplémentaire de raisonnement de logique de premier ordre [3] (plus d'expressivité pour le raisonnement) ou des réseaux bayésiens (gestion de l'incertain).

3.1.3 *Étapes des modèles pour la conscience du contexte*

Des modèles de pensée, tous deux issus de l'informatique ubiquitaire, décrivent les étapes pour un système conscient du contexte de qualité (Fig. 33). Le raisonnement est ainsi séparé en trois étapes [69](Fig. 33a). Une première étape correspond à un traitement de données, à une reconnaissance de motif ou de classes. Puis vient un raisonnement général utilisant entre autres les résultats de la première couche avec l'ajout de connaissances dans des règles ou une ontologie prédéfinies. Enfin les résultats précédents sont utilisés au final à l'intérieur d'une application, via d'autres règles spécifiques ou un langage de requêtes.

Fig. 33b présente une approche hybride idéale [6]. Le modèle propose de manière similaire une première couche de fusion. Puis une différence majeure est que pour faciliter l'efficacité et la mise à l'échelle, le raisonnement principal ne se situe pas directement dans une couche sémantique afin de favoriser l'efficacité. L'application peut directement accéder à cette couche via une interface de type modélisation objet. Il doit cependant donc exister une correspondance entre la couche de raisonnement principale et celle du niveau supérieur pour bénéficier de la réutilisabilité et du raisonnement ontologique. Ce modèle peut ainsi se retrouver dans les environnements sémantiques présentés avec ainsi [56] le moteur de rendu graphique pour le raisonnement efficace, s'interfaçant directement avec l'application, qui bénéficie de moteurs externes et de liens avec la représentation sémantique de la scène en couche supérieure.

3. OWL2 supportant une version étendue de la logique de description, permettant certaines règles, résout partiellement ce besoin d'expressivité du raisonnement au sein de l'ontologie [73]



(a) Étapes classiques [69] (b) Modèle suggéré [6]

FIGURE 33: Étapes pour un système conscient du contexte

Nous aurons ainsi à gérer un protocole d'échange entre l'application et le moteur d'assistance qui correspondra sommairement à la couche ontologique. Des prétraitements peuvent être ainsi fait directement par l'application. Ils sont même essentiels afin d'extraire des informations de haut niveau lorsque l'on veut connecter un moteur sémantique à un environnement qui ne l'est pas. Par exemple, MoBe [28] utilise des règles bas niveaux pour relier les capteurs à une série de concepts abstraits.

3.2 MODÈLES D'ADAPTATION

Ainsi le contexte peut être géré dans une couche supérieure haut niveau et mener à des réactions et des adaptations pour l'application. Les modèles d'adaptations nous permettent alors de mieux appréhender la prise de décisions avant assistance.

3.2.1 *Création d'un processus d'adaptation et assistance*

Une adaptation peut bien souvent être définie via un questionnement de type W5H [39] (Fig. 34). Le contexte apparait explicitement pour répondre à la question *quand*. À nouveau, apparait le lien entre une adaptation automatique et la nécessité de gestion de contexte. La conception du processus générique d'assistance correspond ainsi à gérer les possibilités de combinaisons d'informations entre les différents questionnements. En revanche les liens et le déroulement du processus sont plus généraux dans notre cas.

En effet, une gestion générique d'assistance permet à la fois l'identification de la situation avant de décider d'assistances adaptées. Les descripteurs des blocs ne sont donc pas toujours initialement connus. Ainsi, le processus commençant par le but de l'adaptation ne vise que le déclenchement d'assistance pour une situation connue. C'est le cas par exemple du superviseur de rendu multimodal [12] qui inclus dans la requête initiale la description de la situation, y compris le *pourquoi* implicite qui est de choisir un rendu adéquat à une interaction. Au contraire, dans le cas d'un système qui favorise l'identification comme VR-UCAM [55], le *pourquoi* est le

résultat d'une fusion des connaissances initiales. Le moteur d'assistance générique doit ainsi pouvoir effectuer le processus représenté ici et utiliser des objectifs connus. Mais il doit permettre également d'autres processus afin de répondre à la question globale *comment assister*, qui peut nécessiter l'identification préalable des objectifs.

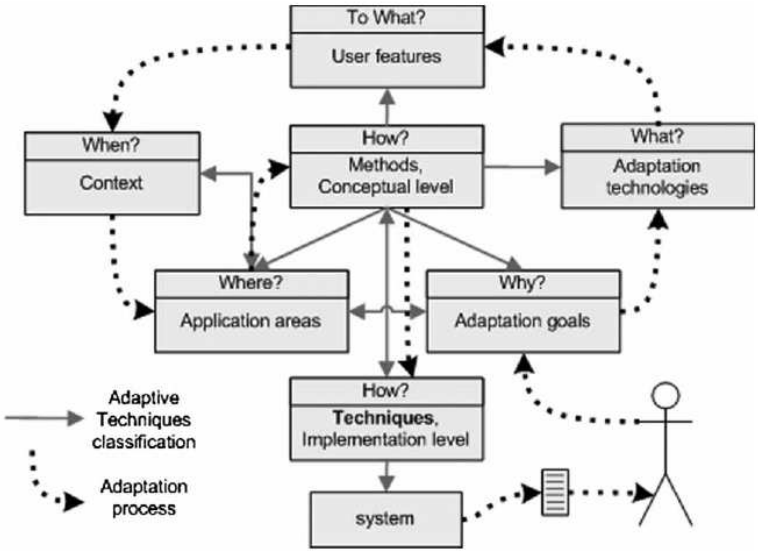


FIGURE 34: Schéma représentant le processus d'adaptation [39]

3.2.2 *Étapes et taxonomies de l'adaptation*

Des taxonomies permettent de distinguer des modèles d'adaptations en fonction de l'acteur réalisant différentes étapes du processus. Ainsi le processus s'étend depuis son Initiation, la Proposition des choix possibles, la Décision parmi ces choix ainsi que l'Exécution finale de ces choix [47]. Les différentes possibilités sont représentées sur un graphe (figure 35) et illustre simultanément les besoins en intelligence pour la compréhension du contexte et du plan (en abscisse) et les besoins pour la présentation de propositions et leur évaluation (en ordonnées). Cette représentation permet d'effectuer un parallèle direct avec les deux tendances du continuum de la conscience du contexte présentées plus tôt (section 2.1.3). Ainsi en abscisse sont représentés les besoins en intelligence en vue de répondre à une problématique d'identification et en ordonnée ceux pour l'assistance. Ainsi six systèmes sont illustrés qui proposent des : [47] :

- adaptations : le système est personnalisable sur intervention explicite de l'utilisateur à toutes les étapes, si ce n'est l'exécution ;
- adaptations initiées par le système : le système propose seul le processus d'adaptation à l'utilisateur, lorsque nécessaire ;
- adaptations aidées par le système : sur demande utilisateur, le système propose des adaptions à l'utilisateur. Celui-ci les choisit et le système les exécute.
- adaptations automatiques contrôlées par l'utilisateur : le système effectue l'ensemble du processus de son initialisation à son exécution à l'exception du choix des adaptations appliquées qui reste sous le contrôle de l'utilisateur ;
- adaptations automatiques déclenchées par l'utilisateur : le système effectue l'ensemble du processus à l'appel explicite de l'utilisateur ;
- adaptations automatiques : le système effectue seul le processus d'adaptations.

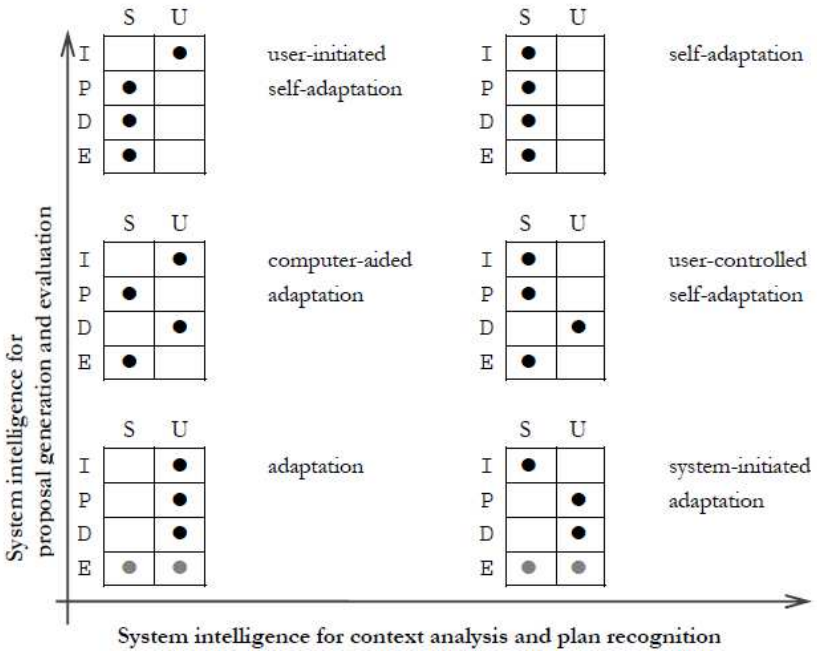


FIGURE 35: Taxonomie des systèmes offrant une forme d'adaptation [47]

Une autre taxonomie (Fig. 36) distingue trois étapes : la détermination des variations potentielles (précédemment la proposition), la sélection des variations effectuées (précédemment la décision) et enfin ajoute une étape de tests de pertinence des variations considérées. Chaque étape peut être effectuée par le concepteur, l'utilisateur ou le système. Les systèmes sont ainsi répartis en six niveaux de sophistication croissante [90] [85] :

- Câblé : le comportement du système est fixé à la conception ;
- Adaptable : le système est personnalisable sur intervention explicite de l'utilisateur ;
- Adaptatif : il sait reconnaître la situation (parmi plusieurs déclencheurs fixés par le concepteur) et applique la recommandation fixée, elle aussi, par le concepteur ;
- Auto-régulateur : il reconnaît la situation et utilise, pour effectuer son choix de réaction, un retour sur lui-même par essai-erreur ;
- Auto-médiateur : autorégulateur mais sait en plus résoudre le problème de l'adaptation par planification et évaluer à priori l'effet d'une réaction ;
- Auto-modificateur : le système, doué de généralisation, est autonome.

Level of system	*Variation*	*Selection*	*Testing*
Designed	*Designer*	*Designer*	*Designer*
Adaptable / tailorable	*Designer*	*User*	*Designer*
Adaptive	*Designer*	*System*	*Designer*
Self-regulating	*Designer*	*System*	*System*
Self-mediating	*Designer*	*System*	*System*
Self-modifying	*System*	*System*	*System*

FIGURE 36: Taxonomie des systèmes d'adaptations [90]

3.2.3 *Discussion sur les étapes de l'adaptation*

Le rôle de l'utilisateur dans les différentes étapes est clarifié par la première taxonomie [47]. Ainsi, pour un processus d'assistance générique devant gérer en pratique tant l'identification de la situation que la prise de décisions des assistances, le système doit proposer des adaptations automatiques complètes. Mais il est intéressant de pouvoir au besoin changer de configuration. Ainsi laisser à l'utilisateur le contrôle de l'initialisation permet de réaliser une assistance universelle que l'on peut ou non réclamer. Le processus d'adaptation peut alors inclure la volonté avérée d'assistance lors de sa prise de décision. Au contraire un système complètement automatique n'est pas sûr de la volonté de changement et le risque de perturber l'utilisateur est plus élevé. De même il est intéressant de pouvoir au besoin laisser la prise de décision à l'utilisateur, parmi une sélection automatique préalable. D'ailleurs, un utilisateur qui possède un contrôle sur le processus d'adaptation ne délègue que peu à peu, selon la situation, de l'autonomie au système [29]. La confiance envers le système adaptatif peut ainsi bénéficier de ces prises de contrôle éventuelles. Ainsi nous souhaitons que le système générique d'assistance puisse être compatible avec ces degrés divers d'inclusion de l'utilisateur.

Un système conscient du contexte doit atteindre au moins le degré adaptatif sur la dernière échelle présentée [90]. Les adaptations potentielles et leur pertinences sont décidées au préalable par le concepteur mais le système décide de leurs exécutions en fonction de la situation. Nous parlerons de méta-adaptation pour les systèmes capables de dépasser l'adaptativité sur cette échelle. En effet, un système capable d'effectuer un retour sur lui-même pour juger de la pertinence de ses adaptations effectue ainsi une adaptation du *processus d'adaptation*. La méta-adaptation pour notre système lui permet d'être potentiellement autorégulateur ou auto-médiateur. Il ne parait en revanche pas possible d'obtenir de notre système qu'il soit auto-modificateur. En effet, il faut pour cela que le système puisse décider lui-même des adaptations potentielles. Or notre système décide d'adaptations pour une application qu'il ne maitrise pas, et ne peut donc ainsi pas les créer de toutes

pièces. Il peut au contraire prendre en compte les possibilités propres de chacune.

Ainsi par rapport aux taxonomie précédentes, une entité supplémentaire peut être introduite : l'application. En effet, le système cité dans les classifications précédentes est à la fois la source et la cible des adaptations. Or la source des adaptations est distincte dans notre approche et correspond à un moteur externe générique d'assistance. La cible principale [4] des adaptations est l'application. Mais l'application peut venir modifier l'étape de proposition du système d'assistance en introduisant un type d'information ou des raisonnements nouveaux. Certains systèmes comme GAIA [72] peuvent tenir compte de cette dynamique alors que d'autres, comme l'approche à l'aide de MBUID [62] fixe l'ensemble obtenu après conception. Les deux pourtant sont des systèmes adaptatifs et proposent des adaptations automatiques pour l'utilisateur. Ainsi nous qualifierons de dynamique l'approche ayant la possibilité de modifier le système au cours de son exécution. Le dynamisme du système d'assistance générique permet de s'adapter notamment aux différentes applications. Mais également in fine de permettre des modifications directes de l'utilisateur [5].

3.2.4 *Positionnement de notre approche*

La conscience du contexte doit permettre une forme d'adaptation automatique. On retrouve donc des points communs dans la modélisation de l'adaptation. Le processus d'adaptation complètement automatique nécessite également deux types de raisonnement : identifier la situation, afin de savoir quand initier le processus et quelles adaptations considérer, puis assister selon la situation, en choisissant les adaptations et en les exécutant. L'éclairage différent de la modélisation de l'adaptation est intéressant. Ainsi le contrôle sur chacune des étapes introduites peut être délégué à l'utilisateur. Pouvoir changer de configuration de contrôle sur ces étapes est un avantage certain à considérer, tant pour le succès de l'adaptation que pour son acceptation par l'utilisateur.

Les systèmes conscients du contexte sont au moins adaptatifs. Mais il est important (comme nous l'avions rajouté dans les propriétés idéales section 2.2.1) de permettre la modification du système d'assistance de manière dynamique par l'utilisateur ou par l'application. Le système peut également se modifier lui-même : c'est la méta-adaptation [6].

3.3 TYPES DE REPRÉSENTATION ET RAISONNEMENT

Nous ne cherchons pas à faire un état de l'art exhaustif des méthodes d'intelligence artificielle applicables au contexte, mais à isoler les caractéristiques souhaitées des approches rencontrées afin de choisir au mieux les formalismes de notre assistance générique. Les méthodes rencontrées pour la réalisation de conscience du contexte sont variées et peuvent être utilisées simultanément. Ainsi MoBe [28] utilise les réseaux bayésiens, l'ontologie et des règles. Une autre approche [31] met

4. Dans le cas de méta-adaptations, le moteur d'assistance s'adapte lui-même.
5. La différence étant la prévision d'une interface spécifique ou un formalisme d'une utilisabilité telle qu'elle soit du niveau de l'utilisateur.
6. Celle-ci devrait être assez naturelle dans les cas où le système permet déjà sa modification dynamique par des entités extérieures.

l'accent sur la fusion de données. Elle utilise ainsi la logique floue pour les mesures capteurs, associées à des concepts d'une ontologie. L'ensemble des capteurs mesurant le même concept est combiné à l'aide de la théorie des croyances, puis un contexte de haut niveau est établi avec des réseaux bayésiens.

3.3.1 *Types de représentation*

3.3.1.1 *Représentation factuelle et représentation sémantique*

Les représentations possibles sont variées. Certaines représentations sont quantifiées et/ou utilisent des concepts de haut niveau. Elles permettent ainsi l'expression d'informations à des degrés sémantiques variés. Les paires attribut/valeur sont efficaces mais n'ont pas de réelle sémantique. Des n-uplets sont parfois utilisés comme des combinaisons d'attribut/valeur et sont présents en logique du premier ordre comme étant un terme associé à une liste finie d'argument. Ces arguments peuvent alors contenir des variables étendant l'expressivité. Les langages à balises introduisent eux une structure fixe qui étend ainsi la sémantique via cette hiérarchie. Les ontologies permettent la définition d'une sémantique riche à l'aide de concepts et de relations ainsi que leurs propriétés.

3.3.1.2 *Représentation graphique*

Les modèles graphiques et les réseaux sémantiques ont différents niveaux d'expressivité mais tendent à être utilisables. Notamment les Graphes Conceptuels (Conceptual Graphs ou CG) qui sont un type de réseaux sémantiques fonctionnels, équivalent à des prédicats de logique du premier ordre interprétés via une ontologie. Les approches graphiques permettent souvent l'encapsulation de représentation. C'est également le principe de l'approche orientée objet qui permet de créer ses propres représentations et d'en cacher la complexité via l'usage d'interfaces.

3.3.1.3 *Représentation de l'incertain et représentation creuse*

Il est difficile de séparer la représentation de l'incertain du raisonnement qui l'accompagne (voir section suivante). La représentation de l'incertain possède souvent une sémantique propre (logique floue, croyance, probabilité). On peut considérer que les méthodes de classification et d'apprentissage comme les réseaux bayésiens (Bayesiean Networks ou BN), les modèles de Markov caché (Hidden Markov Models, HMM) ou les réseaux de fonction de croyance (Evidential Network with Conditional belief functions, ENC) permettent de représenter une situation de manière creuse. Elles ne nécessitent en effet pas de connaissances sémantiques en soi ni même de compréhension particulière préalable. Un modèle initial arbitraire, creux, permet alors de représenter une situation particulière, après apprentissage, à partir d'entrées personnalisées. Ces méthodes permettent une gestion de l'incertitude.

3.3.1.4 *Représentation de processus*

Certaines représentations s'attachent à décrire autant un processus que des connaissances éparses. Par exemple MBUID [62], les réseaux de Petri, les machines d'états finis (Finite State Machine, FSM) et les graphes actionnables en général.

À nouveau, nous sommes à mi-chemin entre le raisonnement et la représentation, dont les graphes contextuels [19] sont un bon exemple. Ils s'attachent à représenter un morceau de contexte prêt à être mis en oeuvre. Proches des arbres de décisions, ils sont plus qu'une variante graphique car la décision n'est pas leur seul but mais également la représentation de différents cas. Ils proposent des embranchements supplémentaires comme les actions parallèles. Le graphe est orienté et son parcours dénote du raisonnement du concepteur en fonction du contexte [19].

3.3.2 *Types de raisonnement*

3.3.2.1 *Logique bi-valuée*

Le raisonnement permet tout d'abord la gestion de vérité, vrai ou faux. Pour obtenir ce raisonnement bi-valué, la représentation via une ontologie utilise le plus souvent une variante de la logique de description (Description logic ou DL). Elle représente une sous-partie décidable de la logique du premier ordre (First Order Logic ou FOL). La FOL est plus expressive et semi-décidable [7]. Elle permet le raisonnement par défaut grâce à la négation par l'absence. La logique d'ordre supérieure est intéressante et permet de raisonner sur des ensembles d'ensembles (sur une réunion de concepts par exemple) mais elle ne propose pas de preuve automatique qui soit effective [8], complète [9] et consistante [10] dans le cadre d'une sémantique standard. Quelques fonctions d'ordre supérieur sont parfois implémentées dans des moteurs du premier ordre (comme *findall* en Prolog).

3.3.2.2 *Logique multi-valuée et gestion des incertitudes*

La gestion des incertitudes ou de la méconnaissance est utile. Un des moyens en est la logique multi-valuée qui introduit des alternatives à la notion de vrai ou faux. La logique floue permet d'aller plus loin en permettant une quantification de l'appartenance simultanée à plusieurs ensembles. Les probabilités et les modèles probabilistes graphiques (BN, HMM, etc.) permettent de quantifier différentes hypothèses et de gérer les incertitudes. La théorie des possibilités et ses graphes (ENC etc.) permettent plus de liberté que les modèles bayésiens dans la répartition des connaissances, avec notamment une gestion de l'inconnu. Les réseaux de neurones sont déterministes mais peuvent également gérer l'imperfection. En effet comme les HMM, les BN ou les méthodes d'apprentissages, ils classent les situations sans avoir de sémantiques explicites, à l'aide de bases d'exemples. Déterminer des critères pertinents en vue d'une classification de situation est le coeur des méthodes de raisonnement par cas.

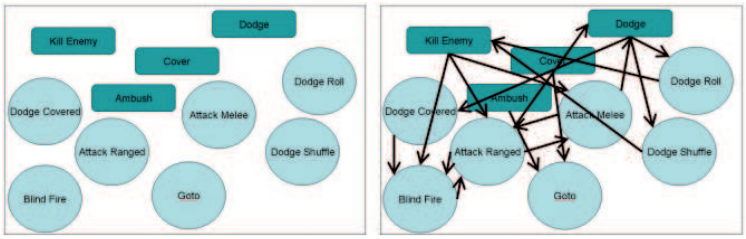
3.3.2.3 *Apprentissage et plan*

L'apprentissage a l'avantage de permettre une adaptation et une adaptivité accrues. De plus il permet d'obtenir des classifications ou des raisonnements difficiles

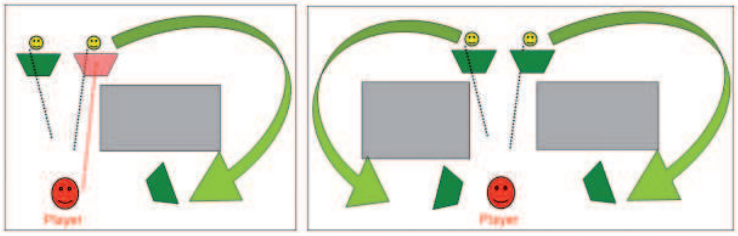
7. On ne peut déterminer à l'avance si un théorème est prouvable au vue des connaissances présentes. Il faut avoir tenter la preuve complète avant de pouvoir répondre par la négative.
8. Permettant de prouver une formule en un nombre fini d'étapes.
9. Toute formule correctement exprimée est démontrable.
10. Les formules dérivées sont toutes vraies.

à établir par le concepteur. Il permet également d'adapter une méthode à un utilisateur, soit de manière complètement automatique, soit en lui demandant d'effectuer des bases d'exemples ou des confirmations (apprentissage par renforcement).

Le planning est la gestion des plans. Le système peut prévoir les conséquences de ses actions et ainsi planifier son approche. Le planning a été intégré à GAIA [72][6]. Il est estimé cependant trop coûteux en temps de calcul et produit bien souvent les mêmes plans. Néanmoins l'approche est intéressante, notamment en dehors de l'informatique ubiquitaire où les capacités de calcul sont limitées. Cela permet de ne pas prévoir tous les cas à l'aide de règles et peut également donner l'illusion d'une capacité de compréhension supérieure à celle qui est réellement prévue. Une illustration peut être faite à l'aide du moteur de FEAR [65]. FEAR est un jeu vidéo opposant le joueur à des personnages autonomes belligérants. En utilisant le planning les embranchements entre modules du raisonnement précédent à base de FSM ne sont plus à faire (Fig. 37a). Le planning gérant chaque soldat ennemi permet de s'adapter au contexte. Ainsi (Fig. 37b) le joueur a l'impression que les soldats tentent de l'encercler, supposant ainsi une réflexion conjointe de ces opposants, alors que chaque soldat ne répond en fait uniquement qu'à son environnement direct (à l'aide d'un planning).



(a) Réutilisation simple de module



(b) Apparition d'une stratégie de groupe non explicitement programmée

FIGURE 37: Intérêt de l'approche planning [65]

3.3.3 *Comparatif des méthodes de représentation et de raisonnement*

Les possibilités de représentation et de raisonnement sont comparées Fig. 38. Les méthodes sont classées selon leur capacité de représentation d'une situation, d'identification de la situation et de raisonnement. Les capacités d'identification et de raisonnement sont liées mais parfois les possibilités d'identification sont intrinsèques à la représentation même (e.g. le classement ontologique) ou représentent l'usage principal des capacités de raisonnement (e.g HMM). Nous avons considéré ici l'apport potentiel de ces différentes méthodes. Nous cherchons à avoir une expression évolutive et riche du contexte, ce qui favorise les méthodes symboliques, notamment via l'usage de concepts. Mais cela ne nous dispense pas pour autant des autres méthodes. En effet, obtenir des possibilités de fusion de données et de

gestion de l'incertain reste important pour un usage durable. Ainsi au choix principal effectué pour notre approche peut être ajouté et/ou intégré l'usage d'autres méthodes. Par exemple celui d'une approche statistique, comme les HMMs, pour obtenir une information haut niveau, par la suite utilisée indépendamment de ce moyen d'acquisition. Le choix de la méthode centrale considérée est présentée section suivante.

Méthodes	Représentation	Identification	Raisonnement	Intérêts
Attributs/Valeur	☺			Efficacité
Modèle à balise	☺			Présentation d'information
Ontologie	☺	😐		Réutilisabilité, interopérabilité
Graphes conceptuels, réseaux sémantiques	☺	😐	☺	Représentation fonctionnelles des connaissances
NN,HMM, BN Apprentissage, etc.		☺	😐	Représentation et raisonnement creux, adaptable
Logique floue Théorie de la croyance Probabilité	😐	☺	☺	Raisonnement à propos de l'incertain de l'imparfait, imprécis etc.
DL		☺	😐	Raisonnement ontologique décidable (Classification de concepts)
FOL et programmation logique	😐	😐	☺	Raisonnement automatique et dynamique
Planning, HTN			☺	Prévision d'état futurs et raisonnement pour un objectif
Réseaux de Petri, graphes contextuels, FSM, MGBUID, etc.	☺		☺	Représentation de processus

FIGURE 38: Comparaison informelle de l'apport de chaque méthode

3.4 CHOIX POUR NOTRE MODÉLISATION

3.4.1 *Représentation et de raisonnement*

Une gestion d'ontologie, afin d'obtenir une sémantique claire pour l'utilisabilité et l'interopérabilité, est à la fois conseillée et l'approche la plus utilisée (section 3.1). De plus, nous aimerions obtenir une bonne expressivité qui est au plus celle de la FOL dans les approches constatées. La discussion sur l'expressivité nécessaire au problème est loin d'être close. Cependant comme le précise le titre du chapitre de "the logic of adaptive behavior", *You Can Only Learn What You Can Represent* [66]. Pour un usage durable et une évolutivité dynamique, l'expressivité est donc une question fondamentale que nous voulons privilégier. Nous avons ainsi choisi de baser notre représentation sur les graphes conceptuels (ou CG). Ils sont naturellement interprétés sur une ontologie. De plus, ils proposent une excellente expressivité (présentés comme une représentation universelle de la connaissance [23][81]). Les CGs ont leurs propres opérations logiques qui peuvent être traduits en termes de FOL. Ils sont cependant plus utilisables car ils sont lisibles par l'homme comme par la machine. De plus ils ont déjà été utilisés (ou leur cousins les réseaux sémantiques) pour des descriptions complexes, notamment pour :

- la personnalisation d'environnement 3D [9] ;
- un environnement sémantique complet [56] ;

– la plasticité d'interface pour l'informatique ubiquitaire [33] ;
– la description temporelle d'un récit [77] ;
– englober les langage à balises [30][78].

Cela permet également de conserver la même sémantique [11] pour le raisonnement comme pour la représentation contrairement aux méthodes hybrides. Les deux peuvent alors être modifiés dynamiquement au sein d'un moteur de logique du premier ordre.

Les réseaux, bayésiens ou similaires, pour le raisonnement se rencontrent également souvent et permettent la gestion de l'incertitude, mais ne sont pas aisés à modifier dynamiquement (la partie centrale de MoBe [28] ou de GULLIVER [41]). Au besoin, il nous est possible d'appeler un module externe au raisonnement principal (e.g HMM, ou un module complet de type MoBe [28]). Cependant il reste nécessaire de gérer un degré d'incertitude au sein de ce raisonnement, que nous intégrerons à la description des graphes conceptuels (comme le fait SOCAM [45] à travers une partie spécifique de l'ontologie).

Ainsi nous obtiendrons un modèle générique, évolutif et utilisable ne privilégiant ni l'identification ni l'assistance. Il peut s'adapter aux besoins de l'application et permettre le raisonnement amenant à des décisions pour l'assistance à l'interaction 3D. Notre approche considérée peut ainsi être comparée aux approches précédentes à l'aide du tableau 39.

3.4.2 *L'ontologie*

L'approche ontologique permet de définir une liste hiérarchisée de concepts et des relations. Plus précisément une ontologie est une spécification explicite et formelle d'une conceptualisation partagée [44]. C'est avant tout une conceptualisation, c'est-à-dire qu'elle permet de construire et de faire référence à un modèle abstrait de phénomènes. La spécification est formelle afin d'être compréhensible par la machine et explicite car elle définit la relation et les contraintes entre les concepts qui constitueront l'univers de l'application. Elle est partagée car elle doit représenter une connaissance acceptée par les membres d'une communauté donnée. L'ontologie nous permet de définir le vocabulaire à partir duquel travailler. C'est une théorie du contenu par rapport à une théorie de mécanismes (règles, réseaux de neurones). Souvent, lorsqu'une théorie du contenu est adéquatement choisie, plusieurs mécanismes peuvent avec une efficacité similaire permettre de répondre à la problématique [22]. La définition de ces concepts est la base de la réutilisabilité et de l'interopérabilité de l'approche sémantique.

Les ontologies peuvent être classées en différents niveaux, pour différentes utilisations. Il est à noter que nous ne cherchons pas à réaliser une ontologie universelle, mais à utiliser l'outil pour répondre à la problématique de l'interaction 3D. En effet, les ontologies à vocation générale divergent dès les tous premiers concepts listés (Fig. 40). Les points récurrents des ontologies sont [22] :
– il y a des *objets* dans le monde.

11. En pratique, les moyens de déductions et de compréhension font bel et bien partie du contexte, par exemple d'une conversation. Nous pouvons alors traiter informations et raisonnements de manière similaire, comme deux éléments d'un même contexte.

TABLE 39: Comparaison des méthodes précédentes et de l'approche considérée

Approche	Représentation	Raisonnement	Description Sémantique	Incertitude	Evolutivité représentation	Evolutivité raisonnement	Utilisabilité	Type privilégié
Technique d'I3D adaptative [11]	-	-	-	-	-	-	Elevée	Assistance
Environnement virtuel sémantique [56]	Svt réseaux sémantiques	Variable	Obligatoire	Variable	Elevée	Elevée	Elevée	Au delà de la conscience du contexte
Superviseur multimodal [12]	FOL	FOL	Non	Non	Moyenne	Élevée	Moyenne	Assistance
VR-UCAM [55]	N-uplet selon le W5H	Appariement	Non	Non	Faible	Moyenne	Moyenne	Identification
GULLIVER [41]	ENC	ENC et filtrage final	Non	Oui	Moyenne	Moyenne statique	Élevée	Conscience du contexte
GAIA [72]	FOL	FOL et BN	Non	Oui	Moyenne	Élevée	Moyenne	Conscience du contexte
SOCAM [45]	OWL et FOL	FOL et DL	Oui	Oui	Élevée	Élevée	Moyenne	Conscience du contexte
MoBe [28]	Ontologie et BN	FOL et BN	oui	Oui	Élevée	Moyenne statique	Élevée	Identification
VR-DeMO [62]	MBUID	Événement-Condition-Action	Non	Non	Élevée statique	Elevée statique	Elevée	Conscience du contexte
Notre projet [35]	CG	CG FOL	Oui	Oui	Elevée	Elevée	Elevée	Conscience du contexte

- les objets ont des *propriétés* ou des *attributs* qui peuvent prendre des *valeurs*.
- les objets peuvent avoir diverses *relations* entre eux.
- les propriétés et relations peuvent changer dans le *temps*.
- il y a des *événements* qui se produisent à différents *instants*.
- il a des *processus* auxquels les objets participent et qui se produisent dans le temps.
- le monde et ses objets peuvent être dans différents *états*.
- les événements peuvent être la *cause* d'autres événements ou d'états, qui en sont les *effets*.
- les objets peuvent avoir des *parties*.

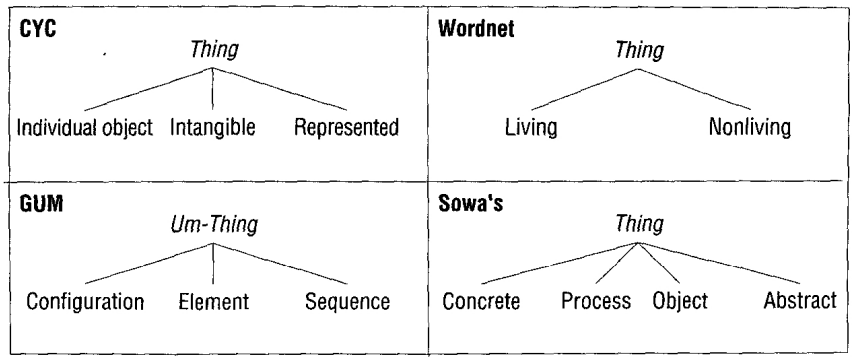


FIGURE 40: Distinction entre ontologies dès le deuxième niveau [22]

Notre ontologie s'inspirera des propriétés communes et nous introduirons peu à peu les concepts nécessaires à notre expression. Le modèle de conception W5H appliqué aux entités, aux situations et aux adaptations, nous permet d'envisager nos besoins d'expression (Fig.41).

Description d'une entité	Description d'une situation	Description d'une adaptation	Questions
Identité	Acteur/sujet	Modules concernés	Qui?
Type	Tâche	Caractéristiques	Quoi?
Usage	Objet/destinataire	Situation	A quoi?
Emplacement	Lieu de la situation	Domaine de l'adaptation	Où?
Dernier/future usage	Moment de l'action	Planification	Quand?
Procédure d'utilisation	Détails de réalisation	Procédure d'adaptation	Comment?
Intérêt ds la situation	Objectif de la tâche	Objectif de l'adaptation	Pourquoi?
Fiabilité	Confiance	Pari	Combien?

FIGURE 41: Un modèle de pensée pour définir notre ontologie en pratique

3.4.3 *Graphes conceptuels*

Le graphe conceptuel (Conceptual Graph, CG) est une représentation graphique et fonctionnelle des connaissances [12]. Ainsi, un CG est une représentation de logique à l'aide de graphes, basée sur les réseaux sémantiques de l'intelligence artificielle [81]. Il permet en pratique de représenter toute situation sous la forme d'un graphe compréhensible par un ordinateur tout en restant également lisible par l'être humain. Pour ce faire, il relie des concepts via des relations. Ainsi le graphe Fig. 42 représente

12. une formalisation détaillée est disponible [23]

une situation. Cette situation possède la logique équivalente à celle représentée Fig.43 mais est beaucoup plus lisible de prime abord (et d'autant plus que les situations sont complexes).

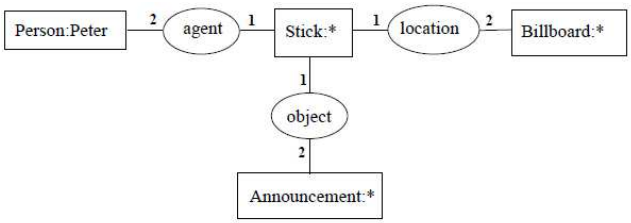


FIGURE 42: Un CG : *une personne, Peter, colle une annonce sur un tableau* [24]

$$\Phi(G) = \exists y_2 y_3 y_4 (Person(Peter) \wedge Stick(y_2) \wedge Announcement(y_3) \wedge Billboard(y_4) \wedge agent(y_2, Peter) \wedge object(y_2, y_3) \wedge location(y_2, y_4)).$$

FIGURE 43: FOL pour : *une personne, Peter, colle une annonce sur un tableau* [24]

Il est possible de spécifier également des graphes emboités. Cela permet de décrire chaque concept d'un graphe à l'aide d'un autre CG, qui se retrouvent ainsi inclus à l'intérieur du premier. Cette notion permet de préciser des informations supplémentaires sous la forme d'un contexte local au concept (Fig. 44). Cette notion est intégrée aux opérations de comparaisons des graphes. Par exemple, le graphes Fig. 44 possède l'ensemble de informations de la situation Fig. 43. En effet, il comprend l'ensemble des branches du graphe précédent, parfois plus spécifique (comme pour la description incluse), ainsi que des branches supplémentaires. Les opérations entre graphes permettent de classer ceux-ci (en effectuant une projection entre les graphes [24]). Finalement cela permet de classer [13] les situations sous-jacentes. La situation Fig. 44 est donc un cas particulier de la situation Fig.43.

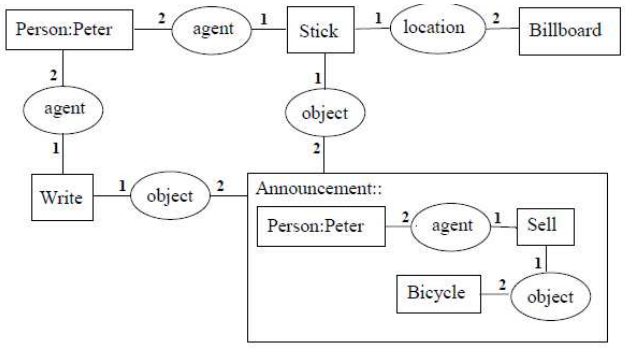


FIGURE 44: Relativité du contexte : *une personne, Peter, écrit et colle une annonce sur un tableau. Cette annonce décrit que Peter vend un vélo* [24]

Ainsi les graphes conceptuels sont naturellement basés sur une ontologie, listant les concepts utilisés. C'est un descripteur universel qui possède l'expressivité équivalente à celle de la logique du premier ordre, avec de plus ces propres opérations. Ces opérations en permettant de comparer deux graphes entres eux

13. D'autres détails sont donnés dans le cadre de l'explicitation des capacités de l'interpréteur Prolog+CG, utilisé pour l'implémentation, section 4.2.2.

permettent finalement de comparer directement deux situations entre elles. De plus ils sont utilisables car faciles à lire par l'être humain tout en étant compréhensibles par la machine. La notion de graphes emboités permet de plus d'introduire la notion de relativité du contexte. Ce sont donc des outils répondant idéalement aux contraintes d'expression du contexte.

3.4.4 *Choix de la plateforme d'implémentation : la plateforme Amine*

De nombreux moteurs implémentent des outils de raisonnement différents [14]. Nous allons nous intéresser uniquement à ceux permettant la gestion de graphes conceptuels (Fig 45). Seules les plateformes Charger [32], Cogitant [42] et Amine [52] gèrent explicitement les opérations algébriques propres aux graphes conceptuels. Charger est principalement un outil de gestion graphique de bases de graphes [15]. Cogitant est en développement constant et est un environnement de développement C++. Amine est une plateforme complète basée sur Java [16]. Les deux dernières sont issues d'une série de travaux sur plus de 20 ans de la part de leurs auteurs. Cogitant semble très bien formalisé et permet une compilation du programme effectué. Amine possède lui un contenu plus conséquent et dépassant la seule définition des CGs. Un de ses avantages est la gestion explicite de l'ontologie ainsi que l'intégration complète des CGs à une couche de programmation logique et un interpréteur de type Prolog : Prolog+CG. La présence de l'interpréteur de logique du premier ordre permet d'avoir une gestion naturelle de la modification dynamique de fait ou de règles.

	Amine	CharGer	CoGITaNT	Corese	WebKB
CG Editor(s)	++	++	++	-	-
Exec. CG	++	+	-	-	-
Algebraic	++	+	+	/	/
KB/Ontology	++	-	?	+	+
Ont. Server	-	-	-	-	++
IDE	++	/	+	-	-
Programming	++	-	-	-	-
Multi-Agent	+	-	-	-	-

FIGURE 45: Comparaison des gestionnaires de graphes conceptuels [52]

La plateforme Amine a donc été choisie pour l'implémentation de notre assistance à l'interaction 3D consciente du contexte. Nous nous intéresserons principalement à la gestion de l'ontologie, des graphes et de la couche de logique (Fig. 46). Mais, elle a de nombreuses autres fonctionnalités qui n'ont pas été abordées. Elles sont néanmoins directement disponibles pour des extensions. Amine offre ainsi certains paradigmes hybrides entre la programmation orientée objet et CG (Par exemple, Synergy a des similarités avec la représentation fonctionnelle de processus).

14. Pour la logique de description : RACER, PELLET etc ; pour les règles : JESS, PROLOG, XSB, etc. ; un mixte : JENA etc. ; pour la logique d'ordre supérieur : SNARK, pour le planning : STRIPS ; pour la création de système expert : CLIPS, pour l'inférence à très grande échelle : OpenCyc ; pour l'intelligence artificielle en générale : SOAR etc.

15. Et sera utilisé pour illustré les graphes de la thèse plus tard.

16. Ces mises à jour sont plus espacées. En revanche il est facile de contacter l'auteur, M. Kabbaj, qui effectue au besoin des extensions et des corrections très rapidement.

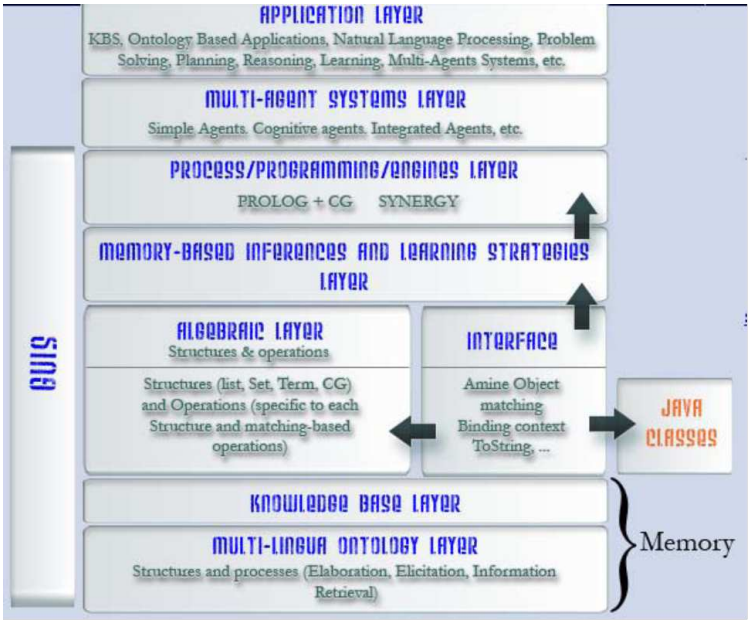
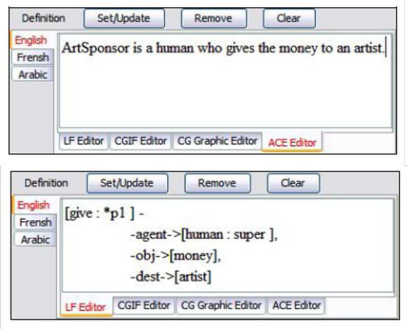
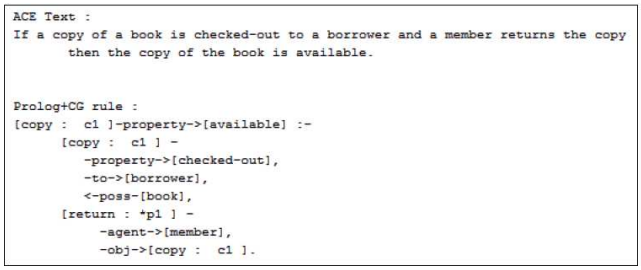


FIGURE 46: La plateforme Amine [51]

La plateforme est activement développée et bénéficiera notamment d'un pont entre le langage ACE (Attempto Controlled English) et les CGs. Le langage ACE est un langage naturel contrôlé. Il permet ainsi d'exprimer un sous-ensemble de phrases en anglais à l'aide d'un vocabulaire et d'une grammaire restreinte. L'interpréteur transforme alors cette phrase pour qu'elle soit compréhensible par la machine et peut désormais mener à des CGs [59]. Le concepteur peut ainsi choisir d'exprimer ces entrées directement à l'aide du langage naturel restreint (Fig.47). Ainsi en perspective, avec la traduction inverse (CG vers ACE), un affichage du contexte par des phrases naturelles sera possible à partir de l'expression initiale en CGs. L'utilisateur pourra donc négocier le contexte naturellement sans pour autant maitriser les CGs [17].



(a) Traduction d'un fait ACE en CG



(b) Traduction d'une règle ACE en règle Prolog+CG

FIGURE 47: Expression de langage naturel traduit en CG dans l'interface d'Amine [59]

3.5 CONCLUSION

Notre objectif est d'obtenir un moteur externe et générique d'assistance à l'interaction 3D. Pour cela une gestion avancée du contexte est nécessaire pour pouvoir

17. Ils restent plus lisibles, notamment sans connaissances a priori, que la logique du premier ordre. Cependant si la compréhension peut être obtenue par tous, avec un effort variable, l'expression n'est pas forcément facile à improviser sans connaissances.

s'adapter tant à la situation qu'aux différentes applications pouvant faire appel au moteur. De très nombreuses approches offrent une gestion du contexte. Mais aucune ne respecte l'ensemble des propriétés idéales et drastiques de la conscience du contexte. En effet le contexte est complexe, dynamique et négociable.

Nous avons alors étudié les modélisations envisageables pour répondre à cette problématique. Le système d'assistance doit ainsi pouvoir identifier une situation et prévoir des assistances. Pour cela, il faut permettre l'accès et la modification de chaque étape de son raisonnement. Par exemple, il doit prévoir les objectifs potentiels en fonction de la description de la situation, mais également pouvoir intégrer un objectif connu, avant de décider des adaptations adéquates. Il a en cela de nombreux points communs avec les différentes formes d'adaptations qu'il doit pouvoir gérer simultanément. Ainsi il offre un processus complètement automatique. Mais il peut laisser à l'application ou à l'utilisateur le contrôle éventuel des différentes étapes de son processus. En permettant ainsi un dynamisme de son contenu, le système peut naturellement se modifier lui-même et proposer des formes de méta-adaptations.

Ces propriétés sont également souhaitables pour se rapprocher de la définition la plus sévère du contexte. Celui-ci émerge alors de l'interaction et est négocié et redéfini à chaque instance par les intervenants qui sont ici l'application, l'utilisateur et le système. L'accès et la modification dynamique permettent à l'application de transmettre de nouvelles informations, possibilités d'adaptation et de nouveaux raisonnements. Cette fonctionnalité permet également, in fine et si besoin est, à l'utilisateur d'interroger puis de mettre à jour l'état du système d'assistance. Enfin elle permet au système via la méta-adaptation de comprendre et de mettre à jour son état lui-même en fonction de son interprétation du monde. La communication est donc potentiellement riche et réellement possible dans les deux sens.

La formalisation d'un tel système n'est pas aisée. La plupart des approches et des modèles utilisent l'ontologie pour obtenir un formalisme extensif, interopérable et utilisable. Le raisonnement dédié à ces concepts est moins expressif que la logique du premier ordre souvent ajoutée et doublant alors le formalisme de représentation. Les approches graphiques sont les plus utilisables mais ne sont pas modifiables dynamiquement. C'est pourquoi les graphes conceptuels, alliant l'approche ontologique, la représentation graphique et une expressivité équivalente et compatible avec la logique du premier ordre, ont été considérés. Ils permettent un unique formalisme tant pour la représentation que pour le raisonnement.

L'implémentation du moteur va alors se baser sur la plate-forme Amine, qui permet la création de systèmes intelligents. Elle gère notamment l'ontologie, les graphes conceptuels et une couche de programmation logique permettant la gestion dynamique tant des éléments d'informations que des raisonnements. En constante évolution, son intégration future d'un langage naturel pour la création ou l'interprétation des graphes conceptuels, permet d'envisager plus concrètement la négociation future avec l'utilisateur sur les bases du travail à venir.

Deuxième partie

CONTRIBUTIONS

"Pa kabab lé mor san éséyé"

Qui se pense incapable, mourra sans avoir essayé

4

CRÉATION DU MOTEUR DE RAISONNEMENT SÉMANTIQUE POUR LA CONSCIENCE DU CONTEXTE

Résumé : *Ce chapitre traite de la création des fondamentaux permettant d'obtenir un raisonnement sémantique externe, qui soit à la fois expressif et modifiable. Tout d'abord est présentée la conceptualisation du moteur permettant d'obtenir ces caractéristiques. Ensuite, la représentation des informations et sa sémantique sont détaillées ainsi que les possibilités de raisonnement offertes par ce choix. Ces capacités de raisonnement ainsi que la sémantique incluse par défaut sont augmentées via la création d'un méta-interpréteur. Celui-ci ajoute notamment une gestion du temps et du degré de vérité. A l'aide de ce canevas pour le raisonnement sémantique, un concepteur peut ajouter ces règles pour obtenir le fonctionnement voulu, comme présenté au chapitre suivant. Cependant, une partie des règles implémentées pour nos applications ont, en fait, introduit des concepts généraux, ici détaillés comme extensions du méta-interpréteur.*

4.1 PRÉSENTATION DU MOTEUR

Le moteur est basé sur la couche de programmation logique Prolog+CG qui gère les graphes conceptuels basés sur une ontologie. Il permet donc une bonne expressivité. A l'aide de ces outils, le moteur doit permettre un raisonnement sémantique, facilement modifiable et utilisable, qui puisse être appelé depuis une application quelconque. Ainsi cette intelligence externe à l'application doit pouvoir obtenir des informations et agir sur l'application. Cette dernière reste fonctionnelle par elle-même et n'est qu'influencée par le moteur (section 4.1.1). Puis, en interne, le moteur doit gérer ces informations et effectuer un raisonnement à la sémantique accessible, selon les possibilités de l'application (section 4.1.2). Enfin, selon les règles implémentées à l'aide du moteur, le raisonnement différera. Cependant il est possible d'envisager une structure d'implémentation préalable. Celle-ci illustre la progression et les types de raisonnements que l'on peut obtenir pour à la fois identifier au mieux la situation et envisager au mieux l'assistance (section 4.1.3).

4.1.1 *Un moteur de raisonnement parallèle*

Le moteur a pour but d'effectuer un raisonnement, utilisant différentes règles, vis-à vis d'une situation décrite par les informations de contexte stockées au moment de la décision (Fig. 48). L'application, siège de l'interaction entre l'utilisateur et le monde virtuel, doit donc pouvoir communiquer avec le moteur tout en restant fonctionnelle seule. Le moteur permet de gérer un ensemble d'influences pouvant affecter l'application dans sa globalité. Ces influences prennent en compte les capa-

cités propres de l'application, fonction des informations fournies et des possibilités d'adaptations offertes. Ainsi, l'application, sans être complètement sémantique, doit être capable de décrire ses informations pertinentes et les moyens permettant de l'affecter directement afin de pouvoir bénéficier d'influences de la part du moteur.

4.1.1.1 *Une communication via des outils*

Ainsi, l'application communique avec le moteur à travers un jeu d'outils (Fig. 48), qui doivent avoir une description sémantique de leur usage afin de pouvoir être déclenchés par le moteur. Les descriptions sont effectuées à l'aide de CGs au sein des outils [1]. Les informations disponibles ainsi que les effets des décisions effectuées sont le plus souvent externes, concernant l'utilisateur, l'interaction en général ou l'environnement virtuel. Les outils de l'application peuvent ainsi être des effecteurs, qui ont un effet perceptible (sur l'environnement, sur l'interaction, en présentant ou en demandant des informations à l'utilisateur etc.) ou des capteurs qui permettent la récupération d'informations (issues de l'environnement, de l'interaction et ou de l'utilisateur). Ces outils peuvent également eux-même être intelligents et avoir leurs propres possibilités de raisonnement indépendamment du moteur (section 5.2 et section 5.3). Ainsi une part de l'intelligence reste distribuée dans l'application même si le traitement du contexte de haut niveau, des informations sémantiques obtenues, est centralisé dans cette intelligence parallèle.

Enfin le moteur peut profiter lui-même des possibilités de modifications dynamiques offertes à l'application et ainsi se modifier lui-même. Des méta-outils ciblant le moteur peuvent donc être également déclenchés. Ils sont séparés en méta-effecteur affectant le moteur en interne (modification de paramètres, de règles etc.) et en méta-capteurs obtenant des informations extérieures à l'application et au moteur, en appelant par exemple un module de raisonnement différent (comme un traitement statistique de données de l'historique).

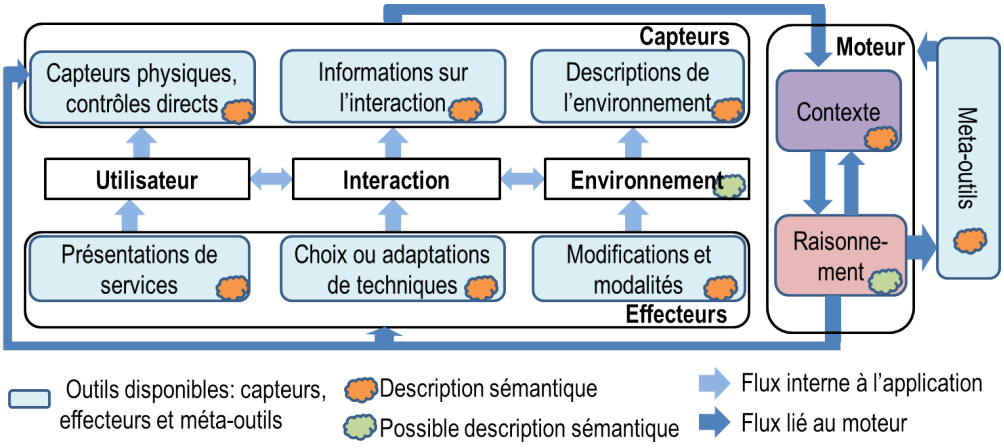


FIGURE 48: Un moteur externe parallèle - communications via des outils sémantiques

Par la suite, les communications avec le moteur sont gérées avec le protocole Open Sound Control (OSC). Les messages pour l'application peuvent être soit centralisés

1. Ces descriptions sont donc stockées ou créées dynamiquement sous formes de chaines de caractères au sein de modules spécifiques de l'application. Il n'est nécessaire ni pour l'application ni pour ces modules d'être capables de les interpréter eux-mêmes.

puis gérés en interne à l'aide d'un protocole propre, soit gérés directement par les outils ayant chacun la possibilité d'une communication OSC.

4.1.1.2 *Les avantages d'une séparation de la réification*

La réification est la mise en oeuvre pratique d'une décision. Séparer cette réification est intrinsèque à l'usage d'un moteur externe à l'application. Mais elle a de nombreux avantages en soi et est même conseillée pour la conscience du contexte :

- Elle permet de redéfinir le lien entre l'action du système et la décision haut niveau issue du contexte. L'application reste ainsi libre à tout moment de la mise en oeuvre pratique de chaque décision sémantique. Par exemple, l'application peut réifier une aide à la visualisation d'une distance à l'aide de l'apparition d'un vecteur, d'une jauge, de couleurs etc.
- Cette séparation permet d'éviter de se soucier, de prime abord, des capacités précises de l'application. En favorisant ainsi l'abstraction, le concepteur s'interroge sur des raisonnements fondamentaux indépendamment de l'application.
- Finalement, cette réification séparée permet une approche composant pour le système d'assistance complet. Les règles du moteur décrivent ainsi des morceaux de raisonnement reproductibles. De même, l'application se voit créer peu à peu une bibliothèque de capteurs et de modules d'assistance réutilisables, souvent indépendamment de l'usage même du moteur.

4.1.2 *Un moteur sémantique*

Les outils réifiant les décisions doivent donc avoir une description sémantique. En effet, le raisonnement au sein du moteur est lui-même sémantique et combine des descriptions à l'aide de CGs d'informations et de règles. Ce raisonnement permet la modification et la compréhension des différentes causalités et résultats intermédiaires. Cela permet au moteur d'être extensif, réutilisable et expressif et bénéficie donc au concepteur. Cela permet également la généricité car l'application peut venir modifier chaque étape mais également questionner de multiples façons différentes le moteur. Enfin, le moteur peut potentiellement gérer des explicitations et ainsi une négociation directe avec l'utilisateur.

4.1.2.1 *Gestion de contexte et de demandes*

Les informations de contexte peuvent revêtir de nombreuses formes (Fig. 49). Tout d'abord l'ontologie liste les concepts et les relations qui peuvent être utilisés par les CGs afin de décrire les règles et les faits. Des CGs peuvent être ajoutés ou retirés dynamiquement. Les modifications typiques concernent des faits et des évènements issus de l'application au cours de l'interaction. Les événements sont des faits spécifiques incluant des informations de temps. La validité des évènements et leur passage dans un historique sont alors automatiquement gérés. Les outils actuellement disponibles et l'historique ont des rôles particuliers mis en valeur.

Le raisonnement est donc influencé par le contexte notamment par les règles décrites. Le processus [2] de raisonnement correspond à la résolution de demandes

2. Ce processus correspond à des usages du méta-interpréteur dont la création est détaillée dans ce chapitre. Il utilise ainsi la FOL en plus de descriptions de CGs. Ce processus d'interprétation fondamentale est donc plus difficilement modifiable. Cela ne limite pas la description par des règles

explicites ou d'une demande de réactions générique. Les réponses à ces deux demandes sont obtenues à l'aide de processus internes [3] permettant la gestion de la vérité et de la confiance qui est un degré de vérité. Eux mêmes sont alors influencés par la gestion du temps affectant l'historique.

Une demande explicite est décrite par un CG tout comme sa réponse. La demande de réactions est elle gérée par un processus supplémentaire qui permet automatiquement d'obtenir les meilleures réactions à appliquer au vue de la situation. Une réaction peut être une question liée à un capteur contrôlé ou une adaptation liée à un effecteur. Cette demande est générique et n'a pas à être décrite. Le processus déclenché récupère l'ensemble des réactions potentielles au vue des outils actuellement disponibles [4] et utilise la notion de confiance et d'impact pour pouvoir les comparer.

La confiance et l'impact de chaque réaction sont donc calculés. La confiance d'une réaction mesure l'utilité probable de son usage actuel. Elle dépend de l'ensemble des chemins de causalités la concernant et de leur confiance spécifique. Globalement, plus il y a de chemins menant à une réaction et plus sa confiance est importante. L'impact est lui spécifique aux réactions et mesure le degré d'effet potentiel sur l'utilisateur de leur déclenchement. L'impact est intrinsèque à chaque décision mais peut être modifié par les circonstances. Une quantité totale d'impact disponible limite les décisions pouvant être prises. Cela permet de ne pas surcharger l'utilisateur de changements au cours de son interaction. Chaque processus élémentaire du moteur est davantage détaillé dans la section 4.3.

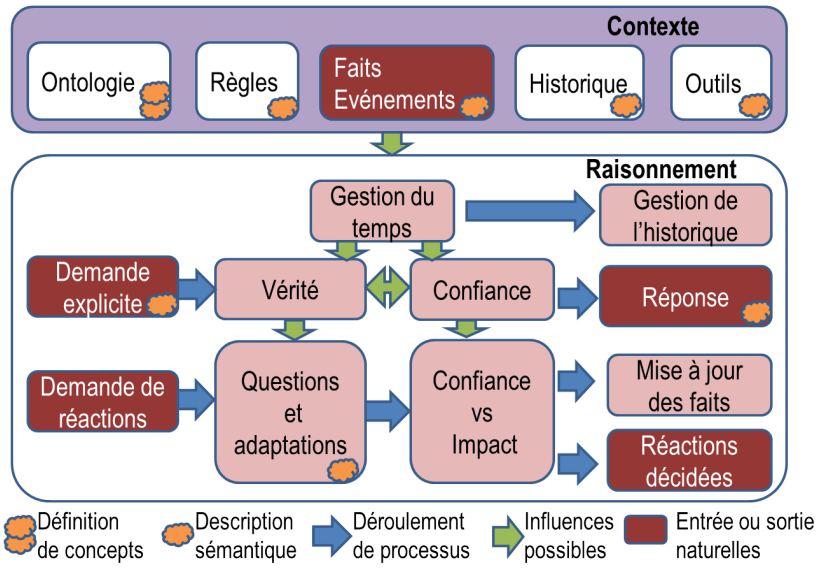


FIGURE 49: Le moteur - gestion du contexte et du raisonnement

CGs d'un raisonnement quelconque sans avoir connaissance de cette couche de FOL. De nombreuses possibilités de configurations via des CGs sont même prévues pour affecter ce noyau.

3. Les processus sont en pratique interdépendants. La demande explicite utilise majoritairement le processus de vérité. Mais obtenir cette vérité peut nécessiter un calcul de confiance, par exemple pour effectuer un choix ou tenir compte de la notion de contraire. La confiance dépends elle bien sûr de ce qui est vrai ou pas.

4. Ce qui utilise le processus de vérité car la présence même d'outil peut être rendue conditionnelle à une situation indépendamment de son usage potentiel.

4.1.2.2 *Accès et évolutivité*

Un des grands avantages du raisonnement sémantique est sa flexibilité et son extensibilité. Ainsi des faits mais également des règles peuvent être ajoutés en cours de fonctionnement. De plus ces ajouts, sémantiques, ne sont pas des fonctionnalités supplémentaires opaques. Conformément à la définition des CGs, ce sont des représentations graphiques fonctionnelles. Le but est ainsi la manipulation des idées même. Ainsi le processus de vérité ou de confiance peut statuer sur n'importe quelle demande. Il est ainsi possible d'accéder et de modifier chaque étape du raisonnement.

Un exemple est plus parlant. Considérons une base de connaissance permettant le changement de technique d'interaction lorsque la tâche est difficile. La tâche y est considérée difficile selon les propriétés de l'objet ciblé. Il est ainsi possible :

- d'ajouter des chemins entièrement nouveaux pour des réactions existantes : e.g. décrire la possibilité de changer de technique d'interaction lorsque celle utilisée n'est pas la préférée de l'utilisateur ;
- d'ajouter des chemins menant à des réactions existantes : e.g. décrire des mouvements de l'utilisateur qui rendent la tâche difficile ;
- d'influencer un chemin menant à des réactions existantes : e.g. statuer directement sur le fait que la tâche soit difficile à partir d'un capteur intelligent ou via un contrôle direct/un profil utilisateur ;
- de s'intéresser aux informations intermédiaires en soi : e.g. ajouter l'affichage sur l'interface utilisateur que la tâche est plutôt difficile [5]. L'utilisateur peut alors prendre seul l'initiative d'un changement de technique ou d'approche ;
- d'ajouter de nouvelles réactions sur un chemin existant : e.g. proposer un changement de technique parmi plusieurs ou effectuer la tâche à la place de l'utilisateur au delà d'une certaine difficulté [6] ;
- de profiter de la comparaison de situations : e.g. une situation décrivant un cas particulier de tâche difficile déclenche également le chemin plus général.
- de profiter de la combinaison d'informations : e.g lorsque les propriétés de l'objet et le profil utilisateur indiquent la tâche comme difficile, la confiance en une réaction appropriée de changement de technique est plus grande. Chaque chemin contribue à la confiance.
- d'ajouter de nouvelles réactions : e.g. rendre une cible de sélection attractive. Les différentes réactions peuvent coexister [7] sans soucis et la réflexion peut donc être conduite par morceaux.

En revanche, si l'on veut conserver ces différent usages, les règles et propriétés ajoutées sont plus complexes à écrire. En effet en établissant un raisonnement sémantique, il faut essayer tant que possible d'anticiper les différents sens que l'on peut donner à la description écrite. Et techniquement, penser à prévoir différentes formes d'appels (annexe A).

5. A l'aide d'un effecteur pour l'affichage et d'un seuil sur la confiance de "*la tâche est difficile*"
6. Comme pour le cas précédent, cela revient à ajouter un effecteur et une règle mais le but est différent : réagir plutôt que comprendre
7. Il existe une dépendance qui est la limitation globale de l'impact. Elle peut inhiber l'usage de certaines réactions moins adaptées au vu des autres disponibles

4.1.3 *Présentation de l'utilisation pratique du moteur*

L'utilisation désirée du moteur créé dans ce chapitre est détaillée dans le chapitre suivant. Ce dernier détaille la construction d'une base de connaissance pour des adaptations de l'interaction 3D. Les effets de cette base via le moteur sont illustrés dans le dernier chapitre sur les applications. Un modèle de pensée de l'usage du moteur peut déjà être présenté ici.

4.1.3.1 *Conceptualisation des fonctionnalités*

Le moteur offre de nombreuses possibilités. Son usage dépend des règles et faits implémentés. Les étapes considérées pour obtenir avec le moteur une assistance à l'interaction 3D peuvent être illustrée Fig. 50. Des éléments de contexte sont décrits a priori ou obtenus dynamiquement par l'application (des connaissances, des intérêts, des objectifs). Ces éléments servent à l'identification d'interprétations possibles de la situation. Ces différentes interprétations influencent l'assistance potentielle à mettre en place (des éléments d'adaptations spécifique selon l'objet ciblé, des situations difficiles reconnues etc.). Éléments de contexte et interprétations correspondent à une volonté de représenter avec précision la situation actuelle.

Mais en partant d'éléments de contexte ou d'interprétations, il est également possible de délimiter l'ensemble des réactions possibles (avec des questions, des adaptations réactives ou pro-actives et éventuellement des méta-adaptations). Cette étape sert à l'identification des raisonnements que l'on peut implémenter au vu de la situation. Les assistances réellement déclenchées à l'issue du raisonnement dépendent des outils présents et de la classification finale.

Ainsi comme désiré, le moteur permet la représentation et le raisonnement à propos du contexte. Il permet également de traiter les problématiques d'identification de la situation et d'assistance au vu de la situation. Les frontières entre les différents blocs et leur placement selon ces deux axes est une conceptualisation des fonctionnalités. Elle aide à l'organisation mais est approximative car le moteur est très ouvert. Ainsi un objectif peut être issu d'une interprétation et une situation haut niveau peut être directement fournie comme élément de contexte. Mais souvent intérêts et objectifs sont à acquérir du comportement dans l'application. De même, il est courant qu'à un intérêt correspondent des adaptations réactives et qu'un objectif offre des possibilités nouvelles d'adaptations pro-actives. Enfin des situations reconnues sont la plupart du temps issues de combinaisons d'éléments sémantiques et donc obtenues au sein du moteur en s'appuyant sur les différents éléments de contexte.

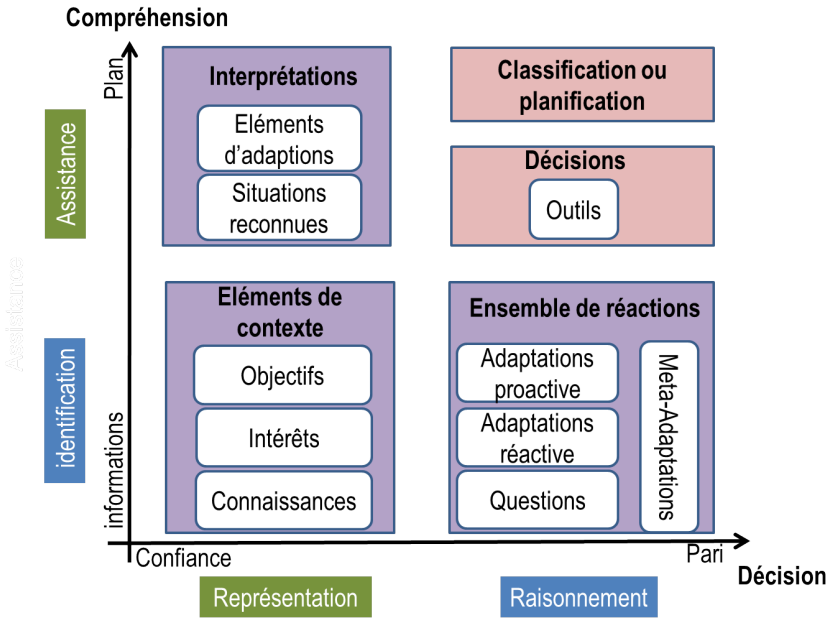


FIGURE 50: Le moteur - Étapes prévues pour l'assistance générique à l'interaction 3D

Le moteur permet donc idéalement de faire progresser la compréhension des informations initiales à un plan décrivant la situation. De même, à partir de confiances initiales le processus de décision progresse jusqu'à effectuer un pari sur les réactions à déclencher. Cette progression est également apparente dans ce manuscrit. L'obtention de contexte est globalement traitée section 5.2. Les intérêts et objectifs de l'utilisateur sont détaillés section 5.3. Des situations particulières sont présentées 5.4. Des éléments d'adaptations sont obtenus, section 5.6. Et enfin, l'ensemble est combiné pour obtenir le comportement d'applications, chapitre 6.

4.1.3.2 *Organisation de fichiers*

En pratique, cette conceptualisation (Fig. 50) apparait plus clairement. En effet, elle correspond à autant de fichiers de connaissances utilisés par le moteur, qui peuvent être chargés séparément. Cette conceptualisation permet ainsi un emploi facilement modulable du moteur par le concepteur de l'application. Elle permet également de mieux appréhender, et donc éventuellement modifier, les interconnexions des différentes règles.

Ainsi plusieurs fichiers définissent les connaissances initiales (la définition de l'interprétation de concepts, les préférences utilisateurs, les spécificités de l'application etc.), d'autres les règles d'identification de situations (e.g. différents états difficiles, disponibles etc.) ou les assistances potentielles (e.g. l'usage de modalités sonores pour un objet non visible etc.)... Ces fichiers ne sont pas directement dépendants. Seuls les fichiers définissant les différents processus du noyau (vérité, confiance, temps et décision) sont fondamentalement interdépendants. Il est possible par exemple de ne charger que le fichier d'assistance. Le moteur ne pourra décider des assistances qu'en se basant sur des informations directement fournies par l'application. En ne chargeant que les fichiers d'interprétations, des informations haut niveau peuvent être récupérées par le concepteur sans déclencher d'assistances.

4.1.3.3 *Résumé de l'implémentation*

La plateforme Amine [52] (une plateforme multicouche pour les systèmes intelligents) est utilisée pour réaliser le moteur sémantique. Elle offre un gestionnaire d'ontologie et un raisonnement de logique du premier ordre de type chainage arrière qui est capable de gérer les graphes conceptuels : Prolog+CG (PCG). Un méta-interpréteur PCG a été écrit pour réaliser le processus de décision décrit dans cette section, gérant une base de connaissance sous la forme de fichiers pré-définis et d'ajouts dynamiques. la plateforme Amine est programmée en Java. L'appel au moteur de décision sémantique se fait donc via un thread Java, capable de communiquer via l'usage d'Open Sound Control (OSC) avec toutes applications supportant le protocole. Nos applications sont réalisées sous Virtools auquel un plugin ajoute la gestion d'OSC.

4.2 LES OUTILS DE REPRÉSENTATION ET DE RAISONNEMENT

Cette section est à mi-chemin entre l'état de l'art et les contributions apportées en discutant de fonctionnement des graphes conceptuels (CG), de la logique du premier ordre (FOL) et des spécificités d'Amine. Elle résume les fonctionnalités par défaut, ainsi que nos usages courants et notre expérience avec ces instruments de travail notamment issue des assistances implémentées au chapitre suivant. La compréhension de leur comportement au sein de la plateforme d'implémentation est une première étape. Ainsi, est présentée premièrement l'apport de l'ontologie en section 4.2.1, puis celui des graphes conceptuels section 4.2.2. Le raisonnement est alors permis par l'interpréteur PCG (section 4.2.3) qui ajoute d'autres possibilités. Basé sur ces fonctionnalités, nous voulons alors introduire à la fois un degré d'incertitude pour la notion de vérité et une gestion du temps. La création d'un méta-interpréteur est alors présentée section suivante.

4.2.1 *L'ontologie*

L'ontologie est de manière basique une liste de termes hiérarchisés. Pour pouvoir raisonner avec des idées et des situations plutôt que des formules, il nous faut gérer un vocabulaire. C'est pourquoi l'ontologie a une importance fondamentale en tant que définition de nos concepts et relations. Lorsqu'un terme fait référence explicitement [8] à son emploi dans l'ontologie, il est dans la suite mis en italique. Les relations diffèrent des concepts par leur emploi futur pour les graphes conceptuels, et sont donc obligatoirement séparées dans l'ontologie.

4.2.1.1 *Notre ontologie*

Ainsi un terme de l'ontologie peut être affilié à plusieurs termes parents. Une liste des relations et concepts est présentée Fig. 51. Les filiations au delà de la première sont définies par un raccourci, représentées par une flèche bleue dans l'ontologie (l'*avatar* est par exemple à la fois un sous-concept de l'*environnement* et de l'*utilisateur*).

8. Souvent l'emploi naturel et l'usage dans l'ontologie coïncident (heureusement !). Seuls les passages spécifiques classant le terme ou son utilisation dans une structure le mettent en italique.

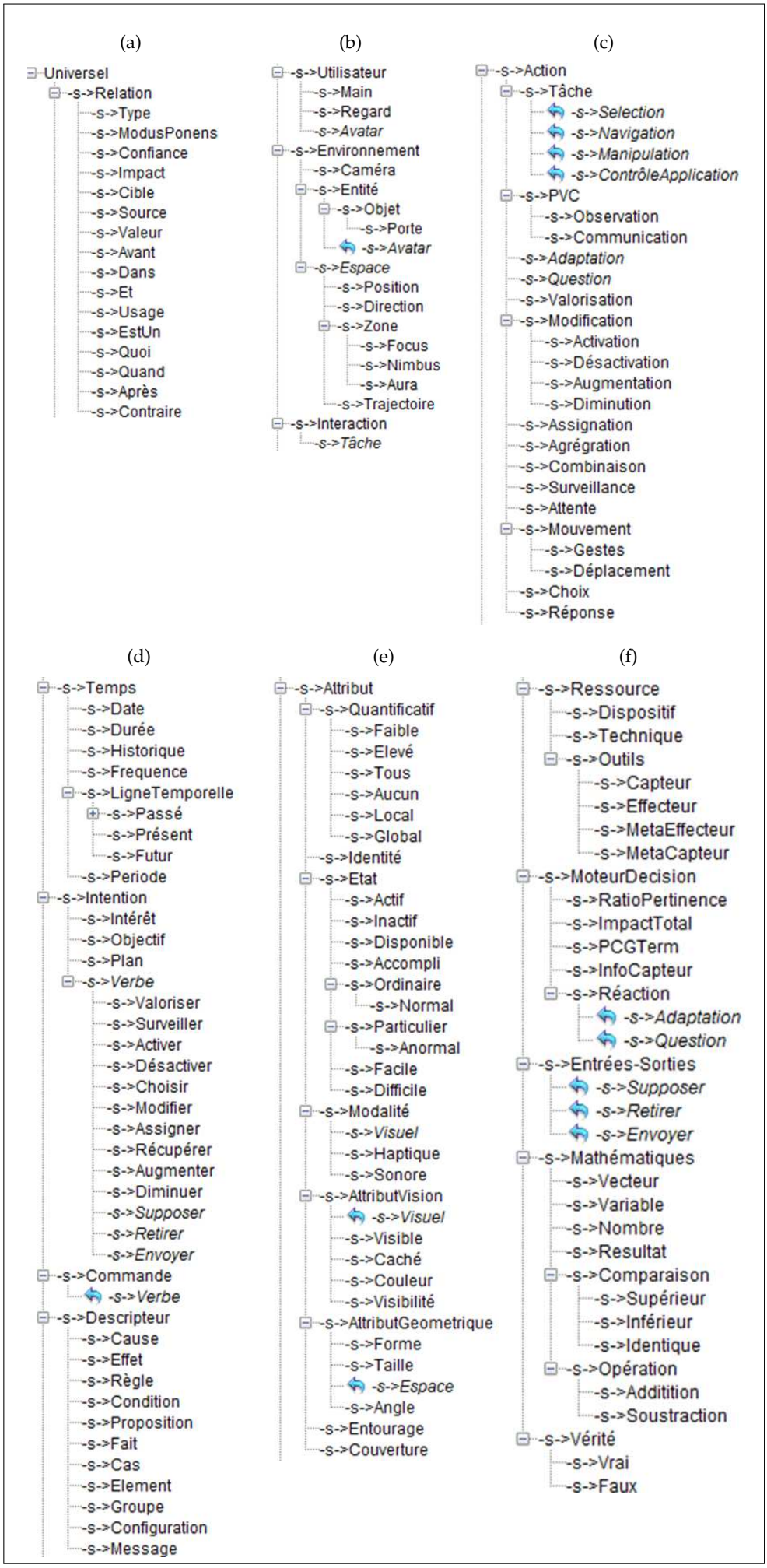


FIGURE 51: Notre ontologie

L'ontologie sépare les relations[9] qui ont un rôle particulier, dont la *confiance* et l'*impact* (Fig. 51a). Puis l'ontologie distingue des classifications pour les concepts d'*utilisateur* en *interaction* avec l'*environnement* (Fig. 51b). Pour pouvoir ensuite exprimer des situations de nombreux autres concepts sont disponibles : des *actions* (Fig. 51c) ; le *temps*, des *intentions*, des *commandes* et des *descripteurs* (Fig. 51d), des *attributs* (Fig. 51e) et enfin des *ressources* avec les *outils* et un ensemble de termes spécifiques dont l'explicitation possible de *vérités* élémentaires (Fig. 51f).

En pratique, la filiation n'est pas le seul lien existant entre termes. Des propriétés supplémentaires peuvent être retranscrites à l'aide de règles ou uniquement dues à une association fréquente dans les CGs. Les *actions* permettent ainsi souvent de décrire l'*interaction* mais ne sont pas limitées à celle-ci et ne sont donc pas un sous-type. Voici donc une taxonomie utilitaire des termes actuellement utilisés :

LES TERMES DE RAISONNEMENT Ils sont utilisés pour statuer de ce qui est vrai, les *faits*, ou de ce qui est juste matière à discussion, les *propositions*. Ils contiennent les *règles* avec des *causes* reliés aux *effets* par la relation de *modusponens*. Également le concept de degré de *confiance* en ces différents éléments. *Causes* et *effets* peuvent préciser des étapes supplémentaires binaires (sans calcul de confiance) : des *conditions* à vérifier *avant* ou *après*. Enfin ce groupe contient également les décisions qui peuvent être prises (des *réactions* comme les *adaptations* ou les *questions*) ;

LES TERMES DE RÉIFICATION Ils sont utilisés pour gérer les *outils*, tels que les *capteurs* ou les *effecteurs*. Leur description utilise les concepts de *commandes* à envoyer pour un *usage* spécifique ainsi que leurs *impacts* dépendants de *cas*. D'autres permettent de gérer les *entités* 3D, la *caméra* et plus généralement les *objets* de l'*environnement* virtuels via leurs *identités* ;

LES TERMES DE TEMPS Ils sont utilisés pour gérer les *faits* nouveaux, les évènements, des *faits* dont une *date* est précisée et pouvant posséder une *durée* (e.g. Fig. 59), également l'*historique* des événements et des *réactions* ;

LES TERMES D'ESPACE Ils sont utilisés pour gérer les *positions*, *directions* etc. Pour les environnement virtuels, la gestion de l'espace est principalement effectuée par le manager de graphes de scènes. Néanmoins pour pouvoir effectuer des descriptions, ces concepts sont nécessaires. De même les concepts de *zones* comme les *auras* ou les *focus* sont utiles pour comprendre l'activité en cours.

LES TERMES D'INTERACTION 3D Ils sont un point central de l'usage de ce moteur autrement générique. Nous avons ainsi besoin de décrire des *modalités* variées, des *tâches*, des *techniques* d'*interaction*, etc.

LES TERMES GÉNÉRAUX Ils forment le vocabulaire de base pour décrire une situation. Par exemple le fait de manipuler des propriétés comme des *attributs*, des *identités* ou l'expression d'*états*, de disposer de *descripteurs* etc.

LES TERMES SPÉCIFIQUES Les applications peuvent également étendre la base de connaissance avec leurs propres concepts. Par exemple au besoin, l'application peut étendre la notions de *gestes*, qui peuvent être nommé ('Z'), ou classés (*droite* et

9. Elles ne sont pas hiérarchisées car la hiérarchie peut être ici trompeuse, section 4.2.1.5. Un concept peut être utilisé comme une relation entre un autre concept et lui même.

haut comme gestes *rectilignes*).

Parmi les types de concepts sont également disponibles [10] avec la couche ontologique d'Amine les individus et les synonymes. Les individus sont des instances d'un concept, qui ne peuvent pas avoir de filiations eux-même. Par exemple, l'ajout des techniques nommées (e.g. "Main Virtuelle") peut se faire comme individus de *technique*. Les synonymes [11] permettent de lier plusieurs termes comme désignant un même concept. Ils favorisent encore la réutilisabilité et l'interopérabilité. La définition même de l'ontologie n'étant pas le but en soit de cette thèse, elle reste perfectible.

4.2.1.2 *Sens donnés aux termes de l'ontologie*

Une partie du sens donné aux termes est directement lié à leur hiérarchisation. Notamment concepts et relations n'ont pas le même emploi au sein d'un CG. Cependant des comportements et des propriétés liés à l'usage de ces termes peuvent être ajoutés. Ainsi des règles donnent en pratique du sens supplémentaire aux termes de l'ontologie (par exemple les règles section 4.5) et permettent une base sémantique plus profonde qu'une simple liste de vocabulaire.

Concentrons-nous par exemple sur deux concepts particuliers, la *confiance* et l'*impact*, (détaillés dans la section 4.3) qui acquièrent leurs propriétés au sein des processus gérant le moteur. La *confiance* représente le degré de vérité d'une information. Pour une *réaction*, elle représente ainsi l'adéquation d'une telle décision au vue de la situation. L'*impact* est spécifique aux *réactions* et représente le degré de répercussion ressentie par l'utilisateur. Ces propriétés donnant un sens aux termes sont ainsi obtenues par la transcription des règles de fonctionnement du moteur .

Ainsi il est possible de donner un sens très spécifiques à des concepts. Cependant pour effectuer un raisonnement sémantique dont on puisse accéder et modifier les différentes étapes, la question de l'expression naturelle [12] se pose. Le concepteur (et à fortiori l'utilisateur) voulant rajouter une règle en se basant sur l'ontologie peut avoir plusieurs interprétations [13] des termes présents. Il est possible de définir des sens multiples afin de favoriser cette expression naturelle (section 4.5.4). Il faut néanmoins définir des conventions d'usages privilégiés qui doivent être naturelles.

Nos conventions d'usages influencent le sens donné aux concepts. Il est parfois possible d'utiliser indistinctement dans le langage courant la notion d'*attribut*, d'*état*, de *type*. Cependant chacun est spécialisé dans le moteur. Ainsi la relation d'*attribut* permet de qualifier une propriété mesurable d'une entité (couleur, position, visibilité etc.). Ils sont notamment obtenus par des *capteurs* (section 5.2). La relation d'*état* permet de qualifier un jugement ou une propriété intrinsèquement transitoire (état actif, déclenché, disponible, anormal, accompli etc.). Les transitions

10. Amine permet aussi des situations, des définitions et des règles qui associent un ou des CGs à un concept. Ces CGs associés doivent par contre être chargés explicitement dans PCG. Ainsi nous travaillons directement sous PCG. En revanche, ils permettent des raisonnements de type explicitation de la part des outils de la couche ontologique à considérer pour un dialogue avec l'utilisateur.
11. Amine permet également une gestion de différentes langues d'une même ontologie.
12. In fine, il faudrait désambiguïser les entrées utilisateur.
13. C'est là toute la difficulté d'une ontologie, notamment universelle. Réussir à introduire une conceptualisation qui soit partagée par le plus grand nombre.

peuvent souvent être gérées par le moteur (section 4.5.2). La relation de *type* permet d'introduire un classement ou d'adresser celui induit par l'ontologie. Enfin on peut indiquer un rôle assumé, indiqué par la relation *estUn*. C'est un qualificatif non mesurable [14] mais qui suppose un certain comportement. Sa durée n'est pas connue, elle peut être indéfinie tout comme transitoire. Un rôle peut être associé à un objet ou une situation (un intérêt, un objectif, etc. section 5.3) et à un individu (débutant, profession etc.).

Les rôles ou les états permettent également d'établir des étapes intermédiaires du raisonnement tout comme le font les volontés d'actions (section 5.4). En effet, l'utilisateur, le concepteur ou l'application peuvent demander au moteur de considérer une *action* particulière. Cette *commande* est exprimée naturellement par un *verbe* d'*action* qui exprime une *intention*. Le moteur, en fonction du reste du contexte concrétisera ou non cette demande. Ainsi la volonté d'action est une étape de réflexion pour le moteur afin de décider de la vérité de l'*action* correspondante. Il reste possible de statuer directement sur l'*action*.

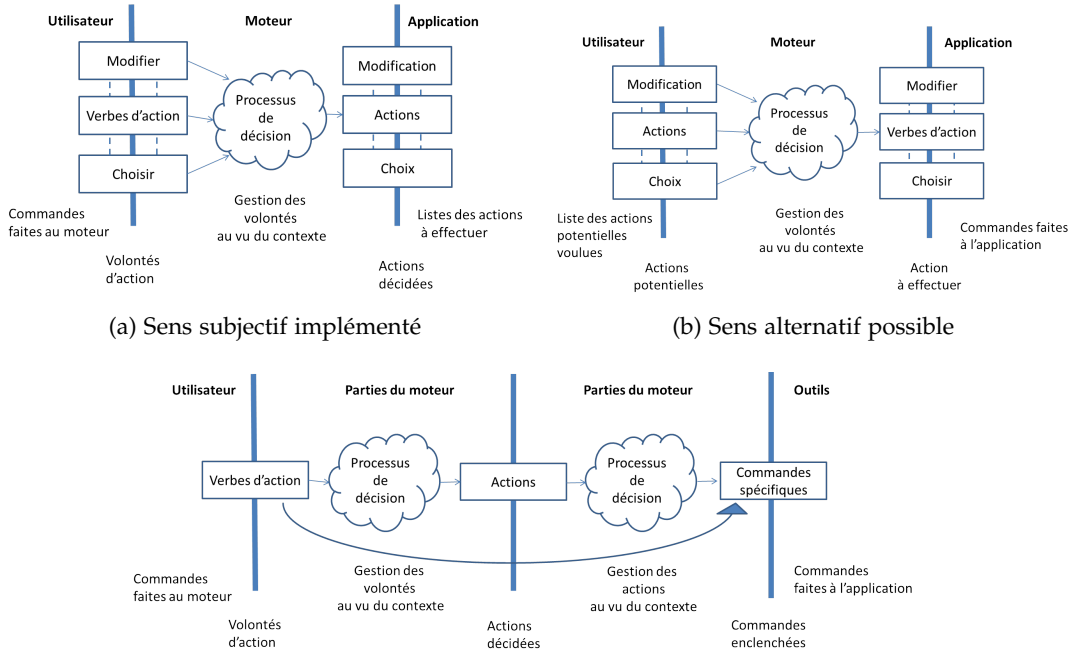
4.2.1.3 *Discussion d'une des conventions d'usages*

Ainsi un verbe d'action ne correspond pas toujours à une action concrétisée. C'est une expression qui semble naturelle. Mais cette distinction reste subjective et pourrait même être inversée ! Dans le premier cas, on considère le verbe d'action comme une commande reçue par le moteur qui gère alors l'action à implémenter (Fig. 52a). Mais il est tout à fait justifiable de considérer au contraire que les actions possibles soient proposées au moteur qui lui émet des ordres via des verbes d'actions (Fig. 52b).

Néanmoins si le premier sens a été choisi, c'est parce qu'entre le moteur et l'application se trouvent en pratique des *outils*. Ceux-ci intègrent déjà leur propre *commande*. Ce choix concernant l'ontologie est donc issu de cette image mentale : le moteur permet un tampon entre les commandes verbales extérieures et les commandes effectivement implémentées décrites par les *outils* (Fig. 52c).

Enfin, ces volontés d'actions ne sont pas toujours externes. Elles peuvent être également issues de règles au sein du moteur et servent notamment d'étape intermédiaire dans le raisonnement. Ces étapes intermédiaires peuvent également devenir une finalité en soi. Ainsi réifier ces volontés d'action par un outil (Fig. 52c) permet par exemple de proposer les actions possibles à l'utilisateur.

14. Mais on peut tenter de le deviner par l'étude du comportement.



(a) Sens subjectif implémenté

(b) Sens alternatif possible

(c) Gain de symétrie par le choix du sens implémenté

FIGURE 52: Sens des verbes et des noms d'actions

Cela illustre malheureusement la subjectivité rapidement croissante de l'implémentation, notamment de l'ontologie. Néanmoins cette convention d'usage des verbes n'est en fait présente que dans les règles créées en vue d'applications liées aux moteur. Notamment pour les règles traitant de l'activation (section 4.5.2.1) et du choix (section 4.5.4.1). La part de subjectivité est alors de plus en plus présente au fur et à mesure que le comportement voulu est ciblé. Mais c'est là également un des avantages recherchés du moteur : celui de pouvoir être utilisé de manière personnalisée. Les règles créées sont réutilisables lorsque l'interprétation initiale est partagée.

4.2.1.4 *Difficulté et évolution constante de l'ontologie*

Établir une liste de concepts suffisamment riche pour permettre une expression correcte sans pour autant multiplier les interprétations différentes est très difficile. Une part de subjectivité apparait souvent. C'est pourquoi les ontologies universelles diffèrent toutes dès le premier niveau. Notre ontologie est d'ailleurs une approche mixte. Si des termes généraux sont nécessaires pour une expression sémantique, les termes sont introduits peu à peu selon nos besoins.

Ainsi cette ontologie n'est jamais réellement fixée. Elle peut être étendue à tout moment pour les besoins d'une application. Mais surtout elle est en constante évolution. En pratique, des erreurs de classification, des précisions supplémentaires, une synthèse menant à un terme plus général et des changements d'approches pour l'expression chamboulent régulièrement la liste de concepts. C'est ainsi que mature l'ontologie (et les règles l'accompagnant) comme les autres théories scientifiques. Ainsi plus celle-ci est avancée, plus les hypothèses et les objets manipulés deviennent complexes mais plus l'expression des théorèmes devient simple et limpide. De la même façon avec une bonne ontologie (et sa bonne utilisation) le nombre de règles est limité et celles-ci sont de plus en plus générales. Elles sont alors également

plus compréhensibles, et donc plus facilement réutilisables et modifiables.

Des indices permettent de savoir que l'ontologie est arrivée à un point où elle devrait être refondue. Par exemple, la multiplication de termes très spécifiques ou composé, apparus originellement pour répondre à un besoin particulier. Par la suite, un usage différent ou meilleur peut les rendre caduques. Dans le cas où le besoin est toujours présent, ces termes peuvent être souvent reclassés, donnant naissance à des concepts généraux supplémentaires. Ils peuvent être également relégués au rôle d'individus ou d'identifiants (une description textuelle libre d'un concept dans un CG et donc absent de l'ontologie).

4.2.1.5 *Influence du logiciel sur l'expression de l'ontologie*

L'usage concret, et ainsi le logiciel, a une influence sur l'expression de l'ontologie. Tout comme une langue a une influence sur la tournure et parfois même sur les idées exprimables. Ainsi le pivot de l'interprétation des CGs d'Amine est la relation. Il ne permet alors pas d'obtenir les mêmes propriétés d'une sous-relation que d'un sous-concept. Cela limite l'intérêt des sous-relations et peut même poser problème quant à l'interprétation attendue d'une expression les utilisant. Cela ne limite pas l'expression mais façonne ainsi l'ontologie et son utilisation.

Par exemple, définissons *père* comme sous-relation de *parent* : l'existence du lien *père* ne permet pas de vérifier l'existence de la relation *parent* directement. En revanche, la définition d'une relation unique *parenté* permet au regard des concepts de *parent* et d'un sous-concepts de *père* d'obtenir cette déduction. L'utilisation d'Amine favorise donc l'emploi privilégié de concepts et sous-concepts avec des relations peu hiérarchisées.

4.2.2 *Les graphes conceptuels*

Un graphe conceptuel est en pratique l'écriture fonctionnelle correspondant à une représentation graphique d'une information. D'une lecture aisée, même sans connaissance de leur grammaire (e.g. Fig. 54), ces graphes sont également très expressifs. Ils relient des concepts entre eux via des relations, librement listés dans l'ontologie. Cela créé une phrase d'apparence codifiée mais compréhensible par l'humain, tout en ayant intrinsèquement une structure qui la rend compréhensible par la machine. Finalement un CG statue sur l'existence d'une situation, via l'existence de relations entre concepts.

4.2.2.1 *Taxonomie de nos graphes conceptuels*

Il n'est pas possible, comme pour l'ontologie, de faire une liste exhaustive de graphes conceptuels utilisant nos concepts. En revanche, il est bien sûr possible de les illustrer, notamment les structures récurrentes. En effet, certaines structures spécifiques sont reconnues par le moteur et acquièrent un sens particulier (l'ajout du sens ainsi que les graphes sont détaillés dans la section 4.3). Cela ne limite pas la description des situations contenues dans ces structures. Ainsi une taxonomie fonctionnelle des usages des graphes conceptuels sépare plusieurs catégories :

FAITS Chaque CG directement exprimé est un fait pour la base de connaissance. Néanmoins certaines structures permettent l'ajout d'informations classiques au

sujet d'une situation (détaillées section 4.3.2.1). La notion de *confiance* peut être précisée comme une connaissance a priori pour une *proposition* ou actuelle pour un *fait*. Les *faits* peuvent être situés également dans le temps avec une *date*, ce qui peut déclencher leur péremption au moment de leur usage.

RÈGLES Diverses formes de règles sont interprétables par le moteur (détaillées section 4.3.2.3). Notamment celles spécifiques au moteur explicitent le lien de causalité, le *modusponens* entre *causes* et *effets*. La notion de *règle* peut également être explicite et préciser alors sa propre *confiance*. Des *conditions* supplémentaires peuvent être également précisées à l'intérieur d'une *règle*. Les *cas* ou les *effets* d'un *outil* sont également proches des règles et spécifiques au processus de réaction.

RÉACTIONS *Réactions*, *adaptations* et *questions* sont des termes de l'ontologie. Mais, on désigne également couramment par réactions les CGs qui permettent de déclencher un *outil*. Ces réactions peuvent être des adaptations et n'ont pas besoin de structures spécifiques. Néanmoins certaines peuvent être définies par convenance en liant une *proposition* via la relation *estUn* au concept d'*adaptation*. En revanche, les questions ont par défaut une description spécifique similaire et mettent de plus en valeur l'information recherchée, contenue dans la description du concept *InfoCapteur*. La déduction à effectuer après le retour d'information de l'application peut être intégrée différemment en fonction du contexte où la question a été posée à l'aide du concept de *réponse*.

OUTILS Des *outils* permettent d'implémenter des réactions. La structure décrit leur *usage*, l'*impact* (a priori) de leur utilisation, la définition de leur propre *commande*, à envoyer via le protocole de communication grâce à leur *identité* qui permet de les contacter.

4.2.2.2 *Sens et grammaire des graphes conceptuels*

Le sens, l'écriture et les opérations entre CGs sont détaillés dans cette partie. Prolog+CG (PCG) permet la gestion de CGs au sein d'une couche de programmation logique et leur ajoute ainsi des propriétés. Un CG permet naturellement une description sur plusieurs plans grâce à d'autres CGs emboités dans ses concepts. La description d'un concept peut être ici faite via n'importe quel objet de PCG : un CG comme précédemment, mais aussi une liste ou une chaine de caractère... Ainsi l'interprétation d'une liste de CGs emboitée dans un concept comme étant une liste de descriptions simultanées peut être ajoutée par le méta-interpréteur (section 4.3.2.4). Un autre avantage de cette fonctionnalité est de ne pas avoir à surcharger l'ontologie, car autrement seuls des termes qui y sont présents peuvent être utilisés par les CGs. Il faudrait alors ajouter chaque entité 3D de l'environnement à l'ontologie pour pouvoir les adresser. Ainsi (acquérir et) utiliser une chaine de caractère quelconque pour décrire un concept, notamment le concept d'*identité* permet de gérer aisément les noms utilisés dans l'environnement virtuel.

Il est parfois nécessaire de faire le lien entre des concepts différents faisant référence à la même entité. Pour cela une même étiquette leur est associée. Ces étiquettes sont numérotées à l'aide de "#" sous Amine (Fig. 53). Des concepts compatibles ou identiques mais non étiquetés sont donc considérés par défaut être des instances différentes. Ainsi, l'étiquette est parfois nécessaire à la sémantique. Mais son usage, plutôt que l'occurrence d'une même variable, facilite aussi grandement la lecture

et permet plus de rapidité dans l'unification. Les variables et l'unification sont introduites dans la section présentant les fonctionnalités par défaut de l'interpréteur PCG (section 4.2.3)

Les CGs de cette section [15] sont écrits avec la notation [16] de PCG (Fig. 53) : un concept apparait entre crochets et la relation reliant les concepts est écrite au milieu du lien directionnel. Le début de la flèche liant les concepts est doublé pour marquer le départ de relations multiples vers des branches différentes, juxtaposées comme dans une phrase avec la syntaxe ",".

```
[Manipulation= Rotation]-
                          - cible -> [Objet # 1] ,
                          - impact -> [Objet # 1]
```

FIGURE 53: Exemple d'écriture linéaire sous Amine d'un CG : une *manipulation* décrite par la chaine de caractères "Rotation" *cible* et a un *impact* sur un même *objet*

Les CGs ont leurs propres outils de comparaison, dont la spécialisation qui est essentielle à l'interpréteur PCG. Un graphe peut être ainsi la spécialisation d'un autre (qui réciproquement en est alors la généralisation). Toute généralisation d'un CG présent dans la base de connaissance est ainsi également vraie. En considérant deux graphes, si chaque chaine de concepts présente dans le premier peut être identifiée dans le deuxième à une chaine plus générale, le premier graphe représente un cas particulier du second. Une chaine est plus générale si elle décrit un cheminement identique entre des concepts identiques ou parents. Il est possible que dans la spécialisation éventuelle, un concept possède une description. Il faut alors également pouvoir trouver une description plus générale pour le concept correspondant dans l'autre graphe. Dans le cas ou le concept n'est pas décrit, il est supposé valide quelque soit la description et est donc également plus général. Ainsi lors d'une comparaison, toute description incluse déclenche également une autre comparaison pour ce graphe emboité. Des exemples sont donnés dans la section suivante.

4.2.2.3 *Vers un usage exclusif de graphes conceptuels : exemples d'expressions*

Tous ceci est plus simple avec quelques exemples. Le CG A (Fig. 54) décrit la situation où il existe un objet qui est un intérêt. Le CG B (Fig. 54) ajoute une *identité* décrite par la chaine de caractère "Door1". La description d'un concept peut être plus élaborée comme celle de la *proposition* d'*usage*. Une *proposition* est un concept englobant générique qui permet de statuer sur une situation complexe. CG C (Fig. 54) détaille encore plus l'*usage* comme la *navigation* vers une *zone* connue comme la "cuisine".

15. Dans les sections suivantes, "CharGer" est utilisé pour illustrer les graphes. Cela les rend encore plus immédiats à la lecture que l'écriture linéaire. Amine propose également un éditeur graphique mais a un rendu visuellement moins intéressant et moins paramétrables.

16. PCG est compatible avec l'écriture standard CGIF mais l'écriture LF d'Amine est plus graphique et également légèrement plus simple que celle originellement proposée par Sowa.

Ces graphes peuvent être comparés. Les graphe A et B ne sont pas des spécialisations l'un de l'autre. En effet, A n'est pas une spécialisation de B, puisque la relation d'*identité* n'y est entre autres pas explicitée. De même, B n'est pas une spécialisation de A puisqu'il statue sur un *objet*, plus général que le type *porte*. En revanche, le graphe C est une spécialisation du graphe B (mais aussi du graphe A) : *Porte* est bien un type d'*Objet* et toutes les branches du graphe B sont également présentes dans une version plus spécifique. En effet concernant les graphes emboités dans les concepts *identité*, la description est identique. Pour ceux correspondant aux usages, la description du concept *proposition* du graphe C est bien également une spécialisation de celle rencontrée dans le graphe B.

```
A= [Porte] - EstUn -> [Intérêt]
B= [Objet]-
        - Usage -> [Proposition=
                [Navigation] - cible -> [Zone] ],
        - EstUn -> [Intérêt],
        - Identité -> [Identité = Dooro1]
C= [Porte]-
        - Usage -> [Proposition=
                [Navigation] - cible -> [Zone] - Identité -> [Identité = Cuisine] ],
        - EstUn -> [Intérêt],
        - Identité -> [Identité = Dooro1]
```

FIGURE 54: Exemples de situations à comparer

Les informations contenues dans les graphes conceptuels peuvent être également écrites avec la logique du premier ordre (FOL) via des quantificateurs d'existence. le graphe A s'écrit ainsi : $\phi(A) = \quad \exists x, y\ \mathit{Porte}(x) \wedge \mathit{Intérêt}(y) \wedge \mathit{EstUn}(x, y)$. Pour gérer les descriptions des concepts, il faut ajouter cette possibilité à l'écriture du premier ordre avec un argument supplémentaire par concept. Les premiers arguments se réfèrent ainsi aux liens entre concepts et le dernier indique l'éventuelle ascendance d'un graphe emboité. Ainsi les graphes Fig.54 peuvent s'écrire Fig.55. A l'aide de ces équivalences, la spécialisation peut s'écrire à l'aide de $\phi(B)$ et $\phi(C)$ les écritures en logique du premier ordre des graphes conceptuels B et C. Ainsi, B est une spécialisation de B car $\phi(C) \Rightarrow \phi(B)$ est vrai. Cette implication peut être vérifiée grâce à la hiérarchie de l'ontologie. Ainsi la parenté entre *objet* et son sous-concept *porte* définit la propriété $\forall x, a\ \mathit{Porte}(x, a) \Rightarrow \mathit{Objet}(x, a)$, quelque soit le lien x ou le concept a.

$$\phi(A) = \quad \exists x, y \; \mathit{Porte}(x, \emptyset) \wedge \mathit{Intérêt}(y, \emptyset) \wedge \mathit{EstUn}(x, y, \emptyset)$$

$$\phi(B) = \quad \exists w, x, y, z, a, b \; \mathit{Objet}(x, \emptyset) \wedge \mathit{Intérêt}(y, \emptyset) \wedge \mathit{EstUn}(x, y, \emptyset) \wedge \mathit{Identité}(x, w, \emptyset) \wedge \mathit{Identité}(w, \emptyset) \wedge \mathit{Dooro1}(\emptyset, w) \wedge \mathit{Usage}(x, z, \emptyset) \wedge \mathit{Proposition}(z, \emptyset) \wedge \mathit{Navigation}(a, z) \wedge \mathit{Zone}(b, z) \wedge \mathit{Vers}(a, b, z)$$

$$\phi(C) = \quad \exists w, x, y, z, a, b, c \; \mathit{Objet}(x, \emptyset) \wedge \mathit{Intérêt}(y, \emptyset) \wedge \mathit{EstUn}(x, y, \emptyset) \wedge \mathit{Identité}(x, w, \emptyset) \wedge \mathit{Identité}(w, \emptyset) \wedge \mathit{Dooro1}(\emptyset, w) \wedge \mathit{Usage}(x, z, \emptyset) \wedge \mathit{Proposition}(z, \emptyset) \wedge \mathit{Navigation}(a, z) \wedge \mathit{Zone}(b, z) \wedge \mathit{Vers}(a, b, z) \wedge \mathit{Identité}(b, c, z) \wedge \mathit{Identité}(c, z) \wedge \mathit{Cuisine}(\emptyset, c)$$

FIGURE 55: Écritures équivalentes en logique du premier ordre

Ainsi, l'expressivité des CGs (e.g. Fig. 54) est en soi mathématiquement équivalente à celle de la FOL (e.g. Fig. 55). Cependant s'exprimer en FOL, notamment pour de larges situations, demande un effort plus important. A tel point que la relativité du contexte visible par les concepts emboités est presque invisible en FOL. Les liens entre concepts sont également plus longs à repérer. Les CGs offrent ainsi une plus grande lisibilité, même sous forme non graphique, et une meilleure utilisabilité. Mais plus encore, leur usage entraine en pratique une expressivité plus importante en permettant des expressions plus naturelles. Nous tenterons ainsi tant que possible de permettre aux concepteurs le paramétrage et la personnalisation de leur usage du moteur à l'aide des seuls CGs, malgré l'usage de FOL dans le noyau, caché, du moteur d'assistance.

4.2.2.4 *Usages et idéal d'expressivité des CGs*

Pour la conception de ce moteur, une des difficultés a été de ne pas limiter ou pervertir l'expressivité initiale. C'est malheureusement nécessairement le cas lors d'un processus de conception. Mais tout particulièrement pour le moteur d'assistance, nous aimerions que sa structure et ses étapes de raisonnement soient idéalement accessibles. Il faut donc que nos usages soient compréhensibles mais également in fine que l'ajout d'une combinaison quelconque issue de l'ontologie, qui ait un sens logique pour un utilisateur, puisse être ajoutée et intégrée par le moteur. S'il est possible d'éviter l'apparition d'erreurs suite à un ajout correct, le moteur ne donne pas toujours au CG le sens que lui prêtait l'utilisateur.

Afin de tendre vers cet idéal, il faut premièrement limiter l'ontologie et ainsi les expressions différentes à gérer, tout en permettant l'écriture des descriptions nécessaires. Des règles d'interprétations sont ajoutées (e.g. la description d'un *fait* est vrai). Cependant la multiplicité des règles d'interprétations ralentit la prise de décision, multipliant à chaque étape les possibilités. Il serait intéressant par la suite d'ajouter une interface afin de traiter et de désambiguïser une entrée utilisateur. Enfin, et c'est le plus difficile, chaque structure ajoutée doit être pensée afin d'être fonctionnelle, combinable et d'éviter les sources d'erreurs.

Par exemple pour décrire un concept, mieux-vaut-il introduire une relation sur le même niveau ou choisir un graphe emboité ? Cela reste subjectif. Dans le premier cas, l'explicitation de la relation est obligatoire dans l'ensemble des CGs qui devront pouvoir correspondre (notion de vérité section 4.2.3). Cela permet par

exemple de limiter les sources d'erreurs. Dans le deuxième cas, cela pose moins de contraintes sur les expressions futures. Cela permet également de lister facilement des informations à l'aide de ce concept englobant (comme les *faits*). En revanche en augmentant les possibilités d'unification, le raisonnement peut être plus lent et, au pire, amener des branchements parasites et des boucles infinies.

4.2.2.5 *Influence du logiciel sur l'expression des CGs*

La théorie générale autorise l'usage des sous-relations pour spécialiser une branche d'un CG, mais Amine ne le gère pas (voir section 4.2.1.5). Ainsi utiliser des sous-relations avec cette implémentation est peu utile et en pratique peut mener à des résultats alors contre-intuitifs. De même, seules les relations dyadiques (liant deux concepts) sont ici autorisées. Cela modèle à nouveau l'expressivité sans pour autant la restreindre car une relation entre trois concepts et plus peut être décomposée via l'usage de multiples relations liant les concepts deux à deux.

Enfin, il est déconseillé d'utiliser plusieurs fois la même relation partant d'un même concept pour bénéficier de l'ensemble des opérations sur les graphes. Cela modèle également l'expressivité puisque la définition de plusieurs qualificatifs via la relation *attribut* dans un même CG est alors déconseillée, e.g le graphe D (Fig. 56). Il faut alors préférer l'usage de relations fonctionnelles, e.g le graphe E (Fig. 56). Une autre solution est d'avoir plusieurs CGs, chacun décrivant un seul attribut avec cette même relation. Cette solution permet de pouvoir obtenir l'ensemble des attributs à partir de la même requête et sera utilisée par la suite. Il faut en revanche exprimer l'*identité* de l'objet dont les attributs sont ainsi séparés.

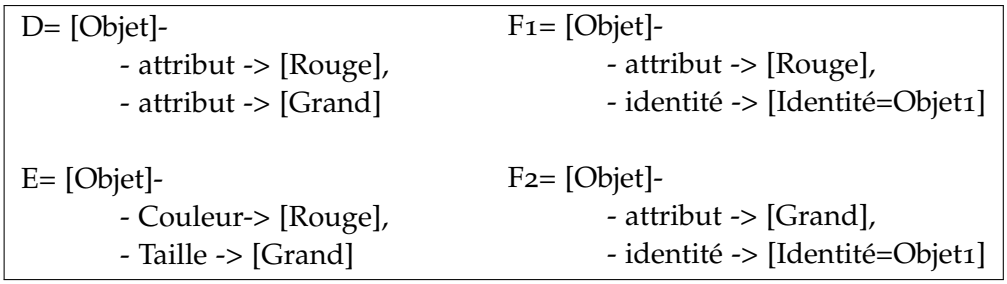


FIGURE 56: Exemples d'expression d'une même situation via des graphes conceptuels

Enfin naturellement deux graphes conceptuels peuvent être une spécialisation l'un de l'autre, et alors dit équivalents, sans pour autant être égaux. La spécialisation est donc un pré-ordre partiel pour l'ensemble des CGs. En effet, les graphes G et H de la figure 57 sont équivalents (chaque branche peut être intégrée à un cas particulier, ou identique, de l'autre graphe). Néanmoins en se limitant aux restrictions conseillées d'Amine (en évitant le graphe H), elle devient une relation d'ordre partiel en acquérant donc l'antisymétrie. Cela nous permet de pouvoir ainsi pouvoir statuer sur l'égalité de deux graphes [17] et ainsi de stocker précisément, au besoin, des résultats intermédiaires [18].

17. ne présentant pas de variables
18. Un des moyens classique d'accélérer Prolog [15].

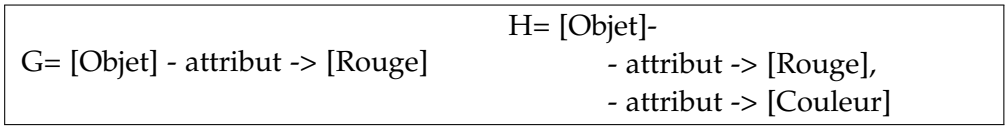


FIGURE 57: Exemple d'équivalence de graphes conceptuels

4.2.3 *L'interpréteur Prolog+CG et ses entités*

Prolog+CG gère de nombreuses entités différentes, dont les CGs basés sur l'ontologie. Un raisonnement peut être mis en place à l'aide d'opérations traitant ces entités, notamment l'unification (notée =) ainsi que des calculs élémentaires. Des expressions, mélangeant opérations et entités, peuvent être reliées à travers des règles. Ainsi l'expression des idées sous PCG dispose :

- de variables libres ;
- d'identifiants simples : une chaine de caractère ;
- d'identifiants avec arguments : une chaine de caractère dénominative, puis entre parenthèses des objets PCG séparés par des virgules ;
- de graphes conceptuels ;
- de listes : des objets PCG ordonnés ;
- d'ensembles : des objets PCG non ordonnés et sans occurrences multiples ;
- de nombres ;
- d'opérations élémentaires ;
- de règles ;

4.2.3.1 *Notion de vérité initiale*

La notion de vérité correspond à la possibilité d'unification [19], vis-à-vis de la base de connaissance, regroupant des expressions liant ces entités. L'écriture d'une requête PCG correspond à l'écriture d'un ou plusieurs termes PCG juxtaposés, qui sont successivement considérés par l'interpréteur. Ainsi pour être vrai, chaque terme de la requête doit pouvoir s'unifier. Pour chaque requête élémentaire R, l'interpréteur recherche un terme X de la base tel que $R = X$. La requête peut est unifiée directement avec un fait PCG (une entité sans conditions). Elle peut également être unifiée au but d'une règle PCG (e.g. Fig. 61a). Dans ce cas les conditions nécessaires pour affirmer le but, doivent également pouvoir toutes être unifiées avec la base de connaissance. L'unification d'un terme avec la base de connaissance est gérée différemment selon que ce terme soit :

- un identifiant simple : le même identifiant dans la base de connaissance ;
- un identifiant avec arguments : le même identifiant avec le même nombre d'arguments, chaque argument devant s'unifier également ;
- un graphe conceptuel : une spécialisation de ce graphe est présent dans la base de connaissance.
- une liste : avec une liste de même longueur, où chaque terme contenu s'unifie successivement.
- un ensemble : avec un ensemble plus grand, où chaque terme peut s'unifier.
- un nombre : avec un nombre équivalent.

19. Une information permettant de respecter l'unification vis-à-vis d'un requête, est souvent désignée par la suite comme une information correspondante.

Une expression utilisant une variable libre s'unifie quelque soit le terme de la base comparée dont l'expression est compatible. Ainsi la variable libre contenue dans une liste peut s'unifier avec n'importe quel terme à la même position dans une liste de même longueur de la base. De même un graphe conceptuel avec un concept libre s'unifie avec n'importe quel graphe de la base, dont le restant de la structure est compatible avec le processus de spécialisation. L'unification peut ainsi se faire par étape, en conservant des variables libres, qui pourront être unifiées au cours du processus. Les apparitions multiples d'une variable dans une même expression est un des liens conservés. A l'issue du processus ces apparitions sont donc toutes unifiées à la même valeur. Si lors de ce processus, l'unification partiellement effectuée n'est plus compatible, le processus revient en arrière et poursuit la recherche (voir Annexe A). Si l'ensemble de la base de connaissance a été parcourue sans avoir pu terminer le processus global d'unification, la requête est évaluée comme fausse. Ce qui permet d'ailleurs l'introduction d'un raisonnement conclusif par défaut (un fait, non présent dans la base de connaissance, n'est pas inconnu mais faux). Ce raisonnement cherchant à établir la vérité de la requête en remontant les conditions nécessaires étape par étape, est appelé raisonnement par chainage arrière.

Par exemple, considérons comme base de connaissance les graphes Fig. 54 et Fig. 56. Les requêtes (utilisant les graphes illustrés (Fig. 58) peuvent directement être entrées par l'utilisateur ou demandées par le moteur, notamment pendant les unifications des causes d'une règle :

- Le graphe I comme requête (Fig. 58) peut s'unifier avec les graphes F1 et F2 (Fig 56). Ainsi la première unification renvoie le couple solution {*Rouge*; Objet1} et la deuxième {*Grand*; Objet1} . Les autres graphes, notamment, manquent de l'une ou l'autre des relations *identité* ou *attribut*.
- La requête correspondant au graphe J peut s'unifier avec les graphes B et C (Fig. 56) et renvoie le graphe L comme première unification, puis le graphe M comme deuxième unification. En effet le graphe C est aussi une spécialisation de J, *porte* étant un sous-type de *objet*. En recherchant des usages d'objets, deux cas particuliers sont trouvés dans la base : le premier pour un objet sans précision, le deuxième pour un objet de type porte.
- K comme requête renvoie M comme réponse. Un seul cas correspondant à l'usage d'une porte a été trouvé.
- La requête composée (et ordonnée) J,K renvoie L comme première unification, puis M comme deuxième unification. L est en effet une généralisation de M, ce qui permet également l'unification de K avec le graphe C, lorsque la variable X est déjà unifiée et égale à L. Il y a en effet bien deux faits correspondant à un objet dont l'usage est précisé. Les deux sont également compatibles avec l'usage d'un objet de type porte.
- La requête composée K,J renvoie uniquement M comme réponse. L'usage d'une porte a été trouvé et est bien compatible avec l'usage d'un objet. En fait dans ce cas particulier où la deuxième condition est une généralisation de la première, celle-ci est superflue : seule la première est limitante.

I= [Objet]-	J= [Objet]- Usage -> [Proposition=X]
- attribut -> [Y],	K= [Porte]- Usage -> [Proposition=X]
- identité -> [Identité=Z]	L= [Navigation] - cible -> [Zone]
M= [Navigation] - cible -> [Zone] - Identité -> [Identité = Cuisine]	

FIGURE 58: Support pour l'illustration des requêtes et de l'unification de CGs

4.2.3.2 *Remarques et influence de l'interpréteur sur les CGs*

Il est important de remarquer que l'unification n'est donc pas symétrique et ne correspond pas à l'égalité mathématique. Pour un CG, la requête cherche conformément à l'intuition à savoir s'il existe une spécialisation, une réalisation particulière du graphe, présent dans la base. En revanche, en écrivant l'opération d'unification soi-même, il faut prendre garde à l'ordre des opérandes. Ainsi en reprenant les exemples Fig. 54, il est vrai que B=C mais faux que C=B. A noter qu'il y a (comme pour Prolog) un blocage de style boucle infinie qui peut apparaitre lorsque le but d'une règle peut s'unifier avec une de ses conditions. Cela implique ainsi qu'une règle ne doit pas être utilisée pour définir directement une spécialisation de ses conditions.

Enfin, une requête initiale concernant uniquement une variable libre, une liste, un ensemble ou un nombre n'est pas possible. Une variable libre pour une relation ne peut pas être évaluée [20]. Un ensemble ne peut jamais contenir de graphe conceptuel, ni d'expression utilisant une variable libre. [21]. Cela façonne à nouveau notre expression et favorise alors l'expression de relations fixes permettant au besoin d'obtenir des concepts variables. Les listes seront également privilégiées aux ensembles.

4.2.4 *Conclusion sur les possibilités initiales*

Les possibilités de représentation et de raisonnement sont très riches. Une situation quelconque peut être représentée naturellement à l'aide d'un CG. Mais de nombreux objets PCG peuvent être présents dans la base de connaissance. Ces objets sont utiles à l'écriture de nos processus mais nous tâcherons de permettre une expression uniquement à l'aide des CGs pour le concepteur et l'utilisateur.

La notion initialement gérée de vrai et de faux est subtile. Elle combine ainsi la théorie des CGs et la FOL. Elle permet l'usage de diverses entités PCG supplémentaires et propose un raisonnement de type chainage arrière. Cela permet l'unification de variables tout au long du processus et donc l'obtention d'ensembles de solutions.

20. Il est possible de l'écrire mais il faut alors que la variable soit unifiée au préalable avant de tenter l'évaluation de cette expression.

21. En effet, un ensemble avec deux graphes pourrait être contenu dans un ensemble avec un seul graphe, dans le cas d'une généralisation commune. De plus, une expression contenant une variable libre peut être évaluée différemment, ne garantissant plus la différence des éléments d'un ensemble.

Nous pouvons alors ajouter nos propres raisonnements. Cependant, au delà de l'expression d'assistances spécifiques, leur gestion générique est souhaitée. Ainsi à l'issue d'un raisonnement sémantique, que peut gérer en soi PCG, un choix entre plusieurs assistances devra être fait. Intégrer une notion de degré de vérité, permettant d'associer à chaque solution une valeur de *confiance*, permettra de gérer cette comparaison finale. Or il faut pouvoir obtenir la *confiance* de n'importe quel raisonnement sémantique ajouté. Ainsi il nous faut intercaler au sein du processus d'unification notre notion de *confiance*. Cela permet alors de bénéficier de la gestion de sémantiques et de variables de PCG tout en obtenant une gestion générique d'une notion d'incertain. De plus l'interaction 3D et les environnements virtuels sont variables dans le temps. Gérer une influence du temps de manière automatique est ainsi également souhaité pour notre assistance. C'est pourquoi la création d'un méta-interpréteur est présentée dans la section suivante.

4.3 CRÉATION DU MÉTA-INTERPRÉTEUR

Le processus de raisonnement s'appuie sur les possibilités initiales de l'interpréteur PCG gérant ontologie et graphes conceptuels. La création d'un méta-interpréteur est alors détaillée afin de pouvoir interpréter des données initiales pouvant intégrer une confiance et un effet du temps (section 4.3.2). Ce méta-interpréteur permet en s'appuyant sur ces formats d'étendre la notion de vérité et d'en intégrer une notion de degré, la confiance (section 4.3.3.1). Il est alors utilisé pour obtenir un processus de réaction générale permettant de déclencher les outils adaptés au contexte (section 4.4). Le méta-interpréteur peut lui-même être modifié au besoin par des CGs (section 4.4.5).

4.3.1 *Motivation de la création d'un méta-interpréteur*

Le moteur doit pouvoir au vu du contexte, décider des réactions les plus adéquates. Comment gérer la prise de décision, tout en essayant de limiter au possible les a priori sur les situations rencontrées ou les adaptations disponibles ? Comment procéder en toute généralité pour des sémantiques différentes ? Une des approches est d'introduire un score pour chaque réaction et de le comparer à un seuil. Toutes les réactions disponibles avec un score suffisant sont applicables. Il faut alors que plus une adaptation est crédible, plus son score soit élevé. Le seuil peut d'ailleurs différer d'une adaptation à l'autre. Les deux doivent être obtenus à partir de la description sémantique. C'est pourquoi les concepts de *confiance* et d'*impact* ont été introduits. La confiance d'une adaptation au vu de la situation, son score, sera comparé à l'impact, au degré d'effet sur l'utilisateur. En effet plus une adaptation aura de répercussion potentielle sur l'utilisateur, plus il convient d'être sûr de soi avant de prendre cette décision.

Or, si l'impact peut être considéré comme intrinsèque, de prime abord, à l'adaptation, la confiance doit être construite au cours du raisonnement. Ce degré de vérité doit être ainsi accessible à chaque étape. C'est aussi nécessaire pour gérer la notion de dynamisme de la base de connaissance, en gérant la notion de temps. Il est possible de raisonner localement au sujet du temps. Mais pour que chaque élément puisse intégrer cette notion et être éventuellement reconnu périmé auto-

matiquement, il faut également pouvoir vérifier à chaque étape d'un raisonnement sa validité. Cependant PCG gère automatiquement par défaut une notion de vrai et de faux. Interférer avec ce raisonnement par défaut pour l'ensemble des CGs nécessite la création d'un interpréteur particulier. Une approche classique en logique du premier ordre est la création d'un méta-interpréteur : un interpréteur PCG, lui-même [22] décrit avec des règles PCG. Plus rapide à implémenter, il est aussi plus lent à l'exécution. De plus le méta-interpréteur n'a pas toujours accès aisément à l'ensemble des informations disponibles. En revanche, l'avantage de cette implémentation est de pouvoir être également modifiée par un raisonnement PCG. Ce qui permet de définir aisément des possibilités de méta-adaptations.

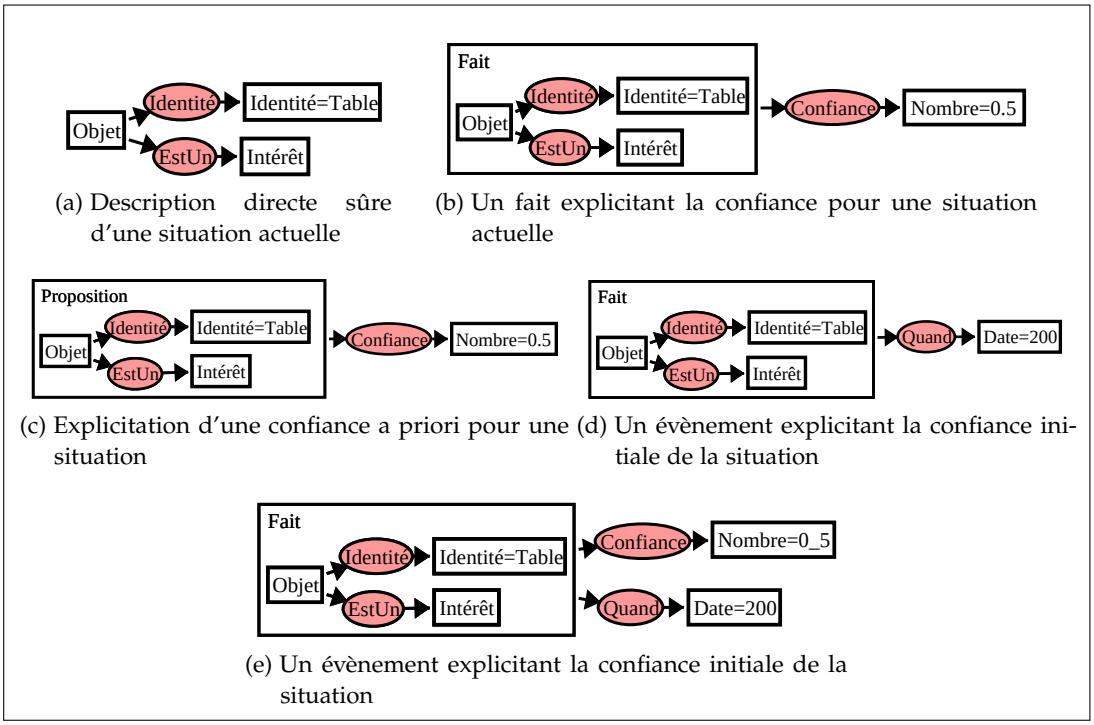
4.3.2 *Extension des informations primaires reconnues*

Il nous faut premièrement être capable de reconnaitre à l'aide du méta-interpréteur différentes informations initiales comme la précision possible d'une confiance (section 4.3.2.1) ou d'une date d'arrivée avec ou sans une durée éventuelle (section 4.3.2.2) pour les faits. La cohérence de ces données initiales va être maintenue et progresser grâce à l'usage de formats de règles personnalisées pouvant elles-aussi préciser une confiance (section 4.3.2.3). Enfin ce format est finalisé et discuté (section 4.3.2.4). Sur ces bases, des processus de gestion de vérité et de confiance peuvent être élaborés. Ils forment, avec la gestion du temps qui leur est intégrée, le méta-interpréteur. Celui-ci, compatible avec l'interpréteur PCG, gère donc nos formats et concepts automatiquement.

4.3.2.1 *Formats d'informations initiales*

Le but est de créer et de gérer automatiquement la notion de confiance et de temps. Il faut tout d'abord avoir des informations initiales qui soient gérées par le méta-interpréteur afin notamment d'exprimer la confiance originelle ou la date d'arrivée d'une situation. Les différentes structures permettant de relier ses informations sont présentées Fig. 59 en prenant l'exemple d'une situation concernant un *objet* d'*intérêt* avec l'identité *table*. Fig. 59a exprime directement le cas où il existe actuellement une telle situation (sans présenter de structure particulière). Fig. 59b exprime la même situation en précisant sa *confiance*. Fig. 59c exprime la connaissance d'une *confiance* a priori à propos de ce genre de situation (sans la supposer pour autant vraie actuellement). Fig. 59d exprime la description d'un évènement (par exemple envoyé par un *capteur*) avec donc une *date* d'occurrence. Fig. 59e précise à cet évènement une *confiance* initiale.

22. Ce qui lui donne ainsi son nom de méta-interpréteur. Cela n'a pas à voir avec la méta-adaptation.



(a) Description directe sûre d'une situation actuelle

(b) Un fait explicitant la confiance pour une situation actuelle

(c) Explicitation d'une confiance a priori pour une situation

(d) Un évènement explicitant la confiance initiale de la situation

(e) Un évènement explicitant la confiance initiale de la situation

FIGURE 59: Exemples de situations et leur confiance

La gestion de ces notions imbriquées dans le processus de raisonnement est l'apport majoritaire du méta-interpréteur. Mais elle ouvre en même temps la voie à l'ajout d'interprétations initiales de la base de connaissance. Les possibilités de vérités élémentaires peuvent être ainsi étendues. En effet, utiliser un *fait* pour décrire une situation autorise implicitement le moteur à interpréter la description de ce concept comme également vraie. Cela permet d'ajouter alors les notions de temps et ou de confiance à une situation, sans avoir besoin de répéter l'information de vérité de la situation dans la base, et donc une expression naturelle. Dans le cas où une notion de temps est ajoutée, la description du fait n'est alors vraie que si l'évènement ainsi représenté est encore valide. L'influence du temps est plus amplement présentée ci-dessous.

4.3.2.2 *Notion de temps*

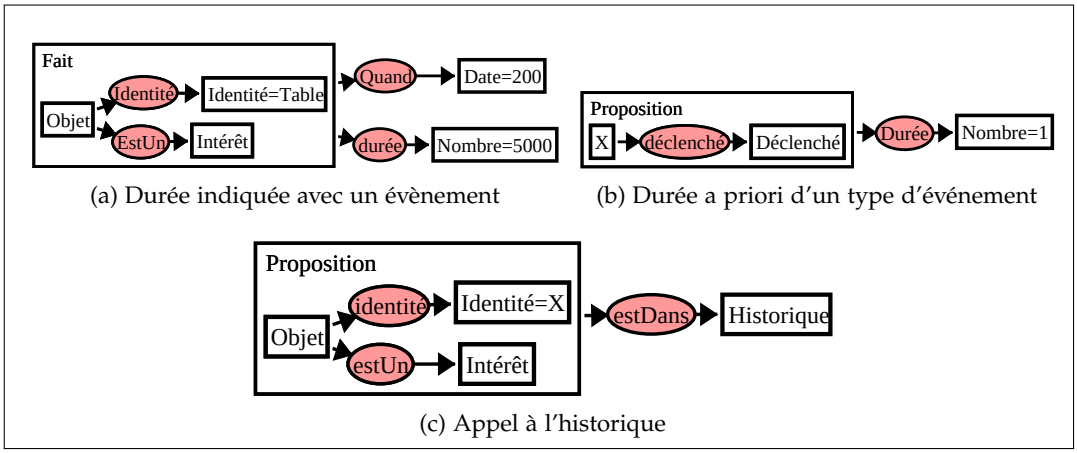
Le contexte va être nourri dynamiquement par des informations issues de l'application. Celle-ci peut ajouter, retirer ou demander directement des informations au moteur. Mais pour permettre plus de liberté aux applications, il est possible que les capteurs ne fassent que rajouter des informations (par choix ou par nécessité). Dans ce cas, ces informations seront probablement périmées à un moment ou à un autre. Il faut alors pouvoir détecter et retirer ces informations anciennes. Ainsi la notion de temps est non seulement une addition appréciable pour l'expressivité mais son ajout permet d'étendre les possibilités d'applications en réduisant les contraintes imposées aux outils.

La description d'un fait peut donc expliciter la notion de date d'arrivée de l'information (e.g. Fig. 59d). Cette date influence la confiance (voir section 4.3.3.2) et la vérité du fait décrit. En effet, le méta-interpréteur[23] vérifie la validité d'un

23. Une autre possibilité pour la gestion du temps est de vérifier avant chaque cycle de réaction l'ensemble des évènements. Cette solution a tout d'abord été implémentée en première approche. Le

évènement avant son usage. Il le retire de la base de connaissance pour le mettre dans l'historique dans le cas où celui-ci est périmé. Pour cela il faut donc connaitre le temps *présent* et la validité du fait. Le temps pourrait être mis à jour directement depuis le moteur. Mais l'application peut avoir son propre déroulement temporel, peut être mise en pause etc. Ainsi l'application doit mettre à jour elle-même l'écoulement du temps via un fait spécifique, la description du concept *présent*. De nouveau, pour pouvoir laisser plus de liberté aux applications, le *présent* se met automatiquement à jour avec l'évènement le plus récent reçu.

La validité des faits peut être paramétrée par la relation *durée*. Les *durées* peuvent être précisées par l'évènement même (e.g. Fig. 60a) ou pour un type d'évènement (Fig. 60b). Les connaissances temporelles a priori se font, comme pour la confiance, à l'aide de la notion de proposition. Par exemple, Fig. 60b décrit l'ensemble des évènements qui se décrivent comme une situation déclenchée (e.g la reconnaissance d'un geste) et leur associe une validité très courte (une seconde). Si à la situation décrite par l'évènement correspond une durée explicite, elle est considérée au lieu de la durée par défaut. Si plusieurs durées sont possibles, le minimum est retenu.



(a) Durée indiquée avec un évènement

(b) Durée a priori d'un type d'événement

(c) Appel à l'historique

FIGURE 60: Exemples de gestion du temps : durée et historique

L'historique est une zone tampon avant l'effacement complet des données. Il stocke les évènements périmés, mais aussi les précédentes réactions. Les faits supprimés sont également mis dans l'historique. L'acte de suppression est alors en soit considéré comme un évènement, daté du moment présent. Cela permet à l'historique de gérer ces données comme des évènements. Les informations de l'historique ne sont alors plus directement consultées, mais peuvent être adressées à tout moment (e.g. Fig. 60c).

L'historique a aussi le même mécanisme de péremption que la base de connaissance classique. La durée par défaut y est plus importante. Il est aussi également possible de définir des type de durées différentes pour l'historique : les tâches actives ou les intérêts précédents peuvent ainsi être décrits comme conservés plus longtemps que les attributs d'objets ou les situations déclenchées.

méta-interpréteur permet de ne faire cette vérification qu'au moment où l'information est nécessaire et permet ainsi de limiter les appels à la base de connaissance.

Enfin, l'historique introduit une gestion de confiance spécifique. L'appel au calcul de la confiance d'un fait dans l'historique (du type Fig. 60c) est interprété comme l'appel à la confiance du fait stocké. Sans cette règle spécifique, la confiance obtenue serait liée au nombre d'apparitions dans la base de l'indexation de la situation dans l'historique. En pratique cette confiance serait alors toujours maximale ou nulle selon que le fait soit ou non présent dans l'historique. Ainsi il devient possible d'ajouter une règle afin de considérer les anciens intérêts comme pouvant toujours être d'actualité (Fig. 92a au chapitre suivant).

4.3.2.3 *Formats de règles*

Pour obtenir les propriétés désirées à partir des informations initiales, il faut avoir accès aux différent éléments à chaque étape de l'unification, pour gérer leurs différentes influences. Or l'ensemble des conditions utilisées ne sont pas accessibles au cours du raisonnement automatique géré par PCG. Par exemple, étudier de manière générique la validité temporelle ou la confiance des conditions d'une règle ne peut pas être imposé aux règles PCG (e.g Fig. 61a). C'est pourquoi ajouter un format de règles maitrisé par un méta-interpréteur est nécessaire pour propager cette notion de degré de vérité tout en vérifiant la validité des faits utilisés. De plus, l'utilisabilité est renforcée car une information élémentaire et une règle se décrivent alors chacune par un unique CG.

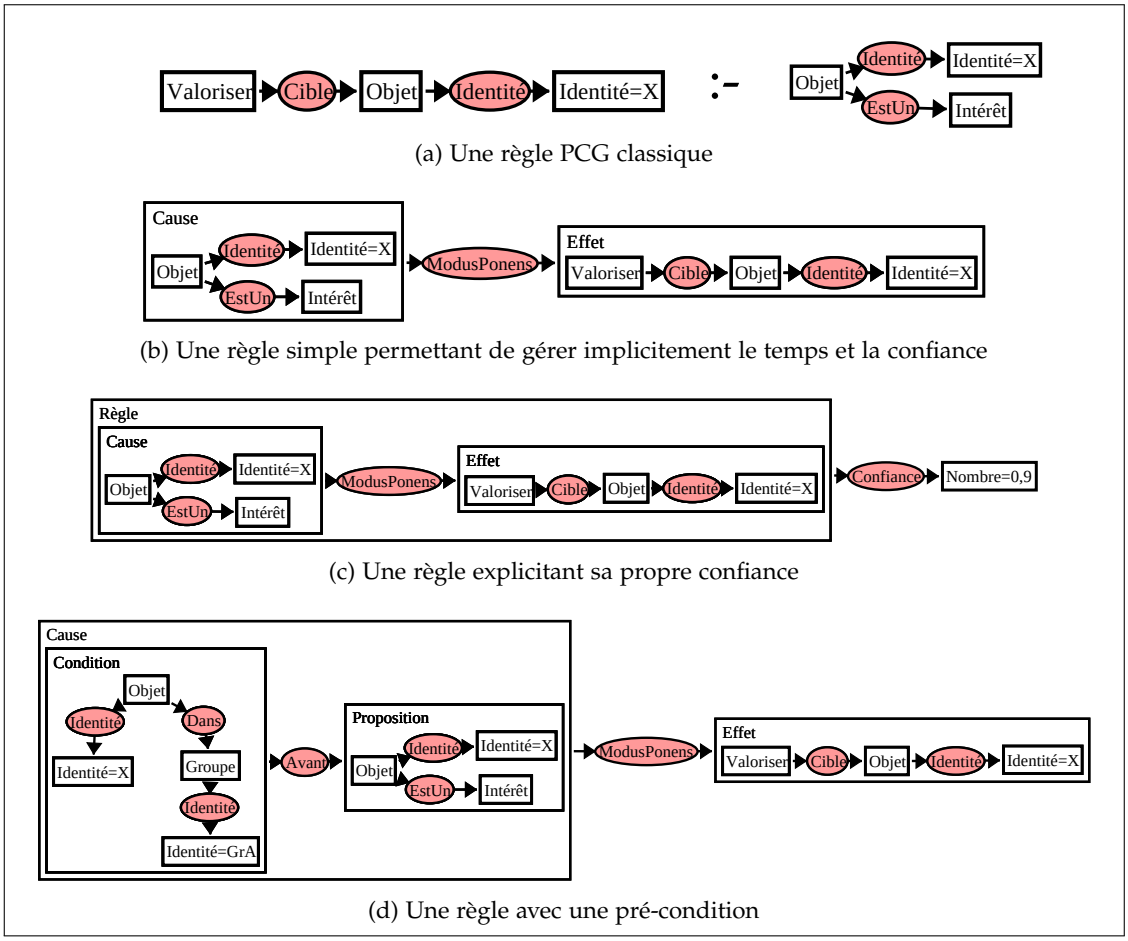
La règle Fig. 61a est ainsi la forme de raisonnement classique de PCG. C'est un lien logique entre deux expressions PCG, ici deux CGs. Ces règles classiques sont également gérées par le méta-interpréteur mais ne sont pas conseillées. En effet, si la règle fait appel directement à des évènements, leur validité ne peut pas être vérifiée. De plus, si l'on recherche la confiance d'une information s'unifiant au but de cette règle, cette possibilité ne renvoi que 1 ou 0, vrai ou faux, quelque soit la confiance des conditions utilisées. Il vaut mieux pour le moteur privilégier l'écriture sous la forme Fig. 61b qui permet le même raisonnement en profitant en revanche automatiquement de la gestion de la validité et du transfert de confiance des causes. C'est la forme la plus simple d'une *règle* spécifique au moteur qui relie les *causes* et les *effets* via une relation spécifique : le *modusponens*.

Le transfert de confiance peut être paramétré, notamment via la *confiance* propre de la règle Fig. 61c. Le produit des confiances des causes est alors pondéré par la confiance de la règle et correspond à la confiance des effets (de cette possibilité). Cette notion est donc bien la *confiance* de la règle : plus cette règle est sûre, plus elle a de poids dans le raisonnement. Avec la possibilité d'une confiance nulle (n'affectant pas le raisonnement) et d'une confiance unitaire maximale (équivalent au cas binaire). Enfin, l'exemple Fig. 61d permet d'ajouter encore un peu de souplesse à nos expressions. Toutes les causes ne se valent pas. Sans pour autant complexifier en introduisant une priorité, les notion de pré-conditions et de post-conditions sont intéressantes à de nombreux égards [24]. Elles permettent de définir des informations à vérifier qui ne joue que sur la vérité mais pas sur la confiance.

Ainsi les causes et les effets peuvent contenir des pré-conditions et des post-conditions. Les pré-conditions des effets sont vérifiées avant [25] de pouvoir considérer

24. Elles peuvent également être utiles pour introduire le planning.
25. Elles représentent donc finalement des méta-causes ayant pour effet la règle elle-même.

la règle. Puis, le méta-interpréteur vérifie que la règle puisse répondre à la requête en cours, c.a.d qu'elle puisse s'unifier avec un des effets. Enfin, les post-conditions des effets sont vérifiés. De manière similaire pour les causes sont évaluées : ses pré-conditions [26], les causes même puis les post-conditions. Elles permettent en pratique de gérer les différents tests logiques nécessaires à l'expression de raisonnements. Ces tests (e.g. l'appartenance à un groupe donné Fig. 61d) binaires renvoie autrement une confiance unitaire, ce qui gonfle donc artificiellement la confiance des effets de la règle.



(a) Une règle PCG classique

(b) Une règle simple permettant de gérer implicitement le temps et la confiance

(c) Une règle explicitant sa propre confiance

(d) Une règle avec une pré-condition

FIGURE 61: Exemples d'expression de règle

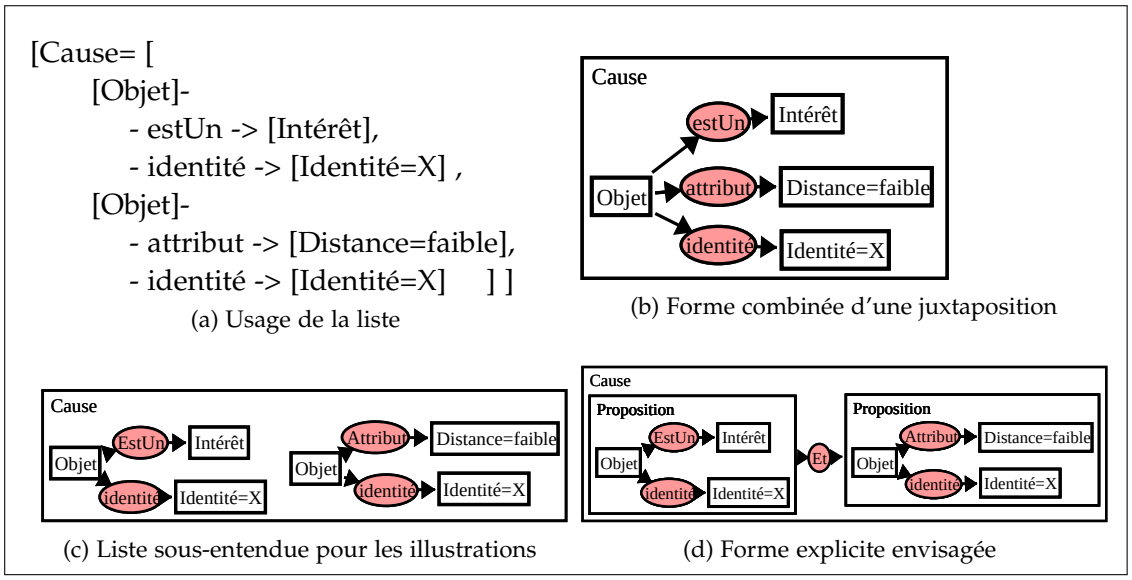
4.3.2.4 *Usages des listes et coordination des CGs*

Il faut pour ces règles CGs pouvoir exprimer et ordonner plusieurs causes voire l'obtention de plusieurs effets. La juxtaposition directes de termes n'est pas permise au sein d'un CG. L'extension naturelle est d'exprimer cette juxtaposition via une liste (e.g. Fig.62a). Il faut alors pouvoir gérer la vérité et la confiance de listes dans nos algorithmes. Par convenance, cette liste même n'est pas représentée dans nos illustration de graphes (e.g. Fig.62c). Une liste d'effets peuvent être aussi précisés et

26. Par volonté de symétrie, l'ensemble des conditions sont gérées bien qu'en pratique une post-condition sur l'effet soit identique à une pré-condition sur la cause (dans le cas d'une interprétation via un chainage arrière).

correspondent à des effets potentiels indépendants [27] les uns des autres.

Pour conserver une syntaxe n'utilisant que des CGs, l'introduction de la relation *et* a été envisagée (e.g. Fig.62d). L'écriture n'est pas réellement plus aisée à comprendre ou à utiliser. De plus, en pratique cela implique pour les processus de séparer les sous-graphes et les stocker dans une liste pour les gérer. La relation *et* n'est donc pas implémentée [28].



(a) Usage de la liste

(b) Forme combinée d'une juxtaposition

(c) Liste sous-entendue pour les illustrations

(d) Forme explicite envisagée

FIGURE 62: Gestion de la coordination additive d'idée

Cependant un souci est rencontré pour les liste de causes et le raisonnement sémantique intuitif. En effet la description combinée (Fig.62b) contient plus d'information que les deux descriptions séparées, même juxtaposées (Fig.62c). Le paradoxe apparent tient au fait que nous interprétons l'identité comme étant unique et un pivot pour réunir les informations. Or un concept quelconque n'a pas cette propriété. Par exemple, s'il existe un objet d'intérêt et un objet éloigné qui sont tous deux des sphères, c'est bien une erreur [29] de conclure qu'il existe une sphère d'intérêt éloignée. La combinaison d'informations n'est ni possible ni souhaitable en toute généralité. La définition de règles de combinaisons (section 4.5.4.2) permet de résoudre partiellement le problème. En revanche, elles sont couteuses en temps de calcul car elles démultiplient les possibilités d'interprétations.

Ainsi en pratique, l'existence de la forme combinée dans la base de connaissance peut déclencher des règles statuant sur des causes décrites avec l'une ou l'autre des possibilités. En revanche l'inverse n'est pas vrai. Il vaut donc mieux pour les règles séparer les informations nécessaires en éléments indivisibles juxtaposés. Cela permet une meilleure compatibilité avec les informations éventuellement disponibles et reflète le fait qu'elles peuvent être obtenues séparément par l'application.

27. Des effets simultanés peuvent être obtenus à l'aide des conditions des causes. Ces effets ne correspondent pas au même paradigme de programmation mais aux étapes d'un processus (annexe A).
28. Elle peut être intéressante avec à nouveau l'ajout d'un traitement préalable des entrées utilisateur.
29. Mais la confiance de l'hypothèse pourrait être renforcée comme dans la théorie des possibilités.

4.3.3 *Extension de la notion de vérité et gestion de confiance*

A partir des formats initiaux d'informations élémentaires et de règles, la création des processus pour obtenir de manière générique la vérité et la confiance d'une description peut être abordée.

4.3.3.1 *Algorithme gérant la vérité*

L'algorithme implémenté pour obtenir la vérité est séparé en deux : le point d'entrée du processus Fig. 63 et le processus principal Fig. 64. Ainsi, l'algorithme traite directement la requête ou successivement ses différents termes dans le cas d'une liste, à l'aide du processus principal. Une liste de termes est donc vraie si l'ensemble des termes sont successivement vrais (Fig. 63).

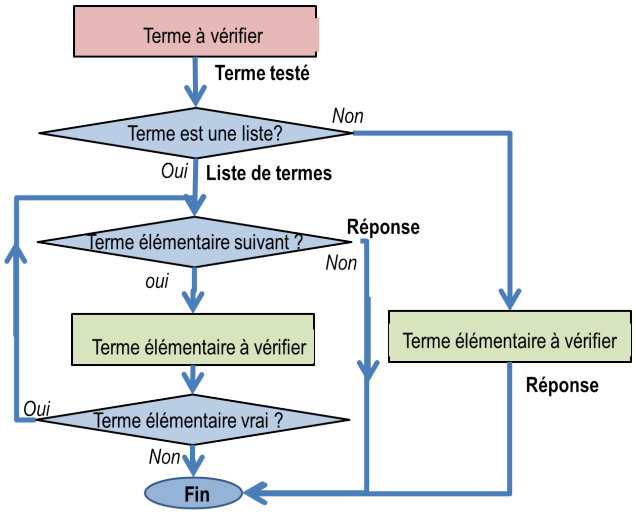


FIGURE 63: Gestion de la vérité

La vérité de chaque terme élémentaire, est obtenue via l'algorithme Fig. 64. Lorsque le terme évalué n'est pas un CG, la notion de vérité est reléguée à l'interpréteur PCG. Dans le cas d'un CG, le processus commence par formater le terme pour extraire la situation élémentaire. Cela permet de gérer les requêtes qui seraient déjà exprimées dans un format d'assertion (e.g. extraire la description d'un *fait* n'introduisant aucune relation). Puis deux blocs se succèdent. Premièrement la gestion des assertions qui recherche la présence de l'information directement dans la base de connaissances, sans utiliser nos formats de règles. La recherche commence avec l'information primaire puis éventuellement l'information sous un des formats spécifiques précisés (e.g. la description d'un *fait*). Si un résultat est trouvé, le processus s'interrompt et le renvoie.

En revanche si aucun des formats ne renvoie un résultat, le bloc gérant les règles est abordé. Celui-ci cherche une de nos règles pouvant mener à la requête. Si une règle est trouvée, ces causes sont relevées (ainsi que les conditions éventuelles) et leur vérité est alors testée. Ce bloc appelle à nouveau le processus complet (Fig. 63) avec une nouvelle entrée : les causes (et conditions). Il diffère ainsi de l'assertion, car une autre requête représentant une situation différente peut être testée, avant de pouvoir répondre à la requête initiale. À nouveau, dès qu'un résultat concernant la requête initiale est trouvé, le processus s'arrête.

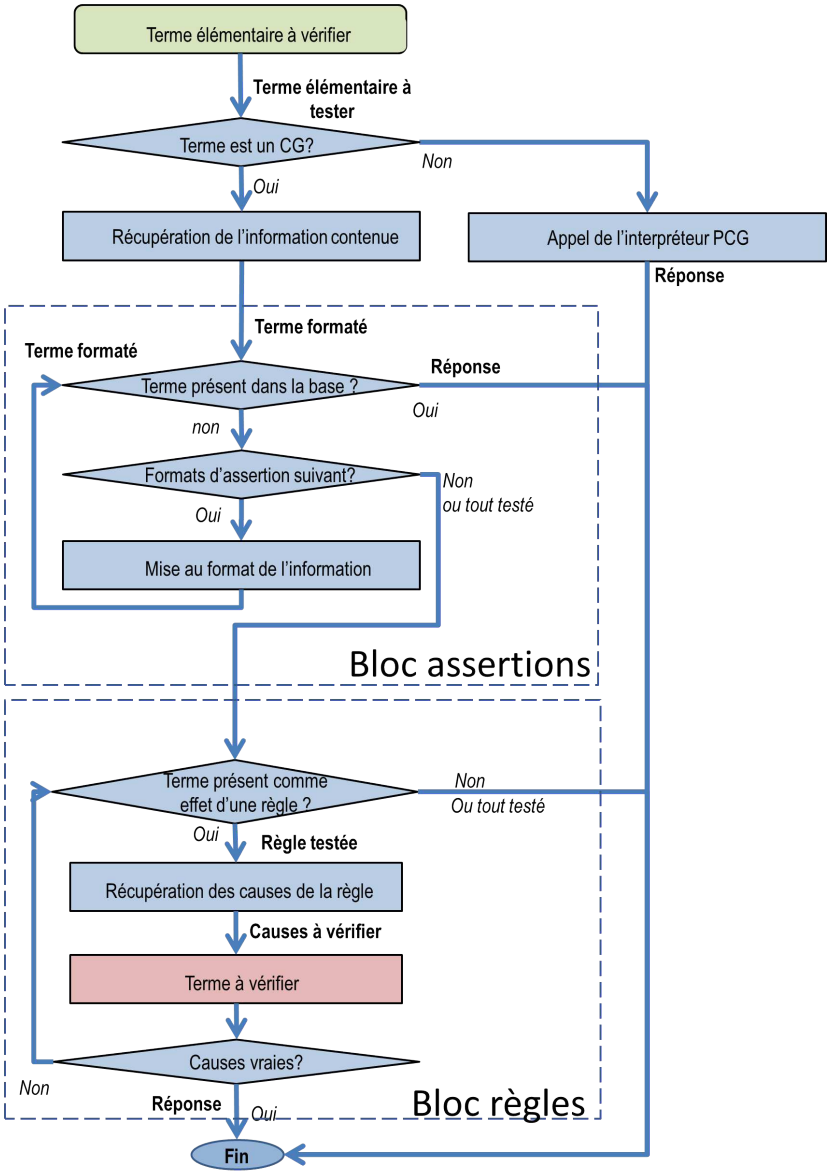


FIGURE 64: Gestion de la vérité d'un terme élémentaire

A l'aide de cet algorithme, la vérité et les solutions correspondant à une requête peuvent être trouvées. La notion de vérité est compatible avec la notion initiale et cohérente avec nos formats. Ainsi un *fait* est vrai, à moins que la gestion du temps associée automatiquement à son format précisant une *date* d'arrivé ne le repère comme périmé, et nos règles sont utilisées dans le raisonnement.

4.3.3.2 *Calcul de la confiance d'un terme*

Le méta-interpréteur va être capable de relier entre elles les notions de confiances dans le raisonnement. Il faut au préalable définir la valeur des possibilités de confiance initiale et expliciter la gestion des confiances multiples associables à une même description. Ainsi pour tout CG, notamment une réaction, une liste des *confiances* est obtenue en considérant l'ensemble des possibilités influentes dans la base de connaissances. Cela peut être :

– un fait correspondant directement dans la base de connaissance PCG (e.g. Fig 59a). Par défaut et compatible avec le raisonnement binaire classique, la confiance est alors maximale égale à 1 ;

– un CG précisant la confiance d'un fait correspondant (e.g Fig. 59b) ;
– un CG précisant la confiance a priori en un fait correspondant (e.g. Fig. 59c) ; C'est le seul cas qui n'assure pas également la vérité du fait. Les confiances a priori sont donc considérées dans la confiance globale uniquement lorsque la vérité est avérée.
– un évènement précisant des informations temporelles vis-à-vis d'un fait correspondant (e.g. Fig. 59e). C'est une confiance qui dépend du temps. Sa confiance initiale est multipliée par le ratio de sa validité restante :
‘

$$1 - \frac{Date\,actuelle - Date\,d'arrivée}{Durée\,de\,validité} \tag{4.1}$$

– une règle précisant un effet correspondant (e.g. Fig. 61). Si la règle est vérifiée, la confiance des effets notamment celui recherché est alors le produit [30] de la confiance des causes pondéré par la confiance de la règle même. C'est donc un processus itératif, puisque le calcul de la confiance des causes appelle l'ensemble du processus à nouveau.

Une fonction est ensuite utilisée pour convertir cette liste en un seul scalaire. Plusieurs fonctions peuvent être choisies : une version optimiste avec le maximum, une version pessimiste avec le minimum, une version représentant l'ensemble des données avec la moyenne, etc. Au final, une fonction de fusion est choisie afin de permettre de refléter non seulement la valeur numérique de l'ensemble des données, la moyenne, mais également le nombre de possibilités menant à l'information recherchée. Ainsi plus il y a de règles et de faits menant à une information (e.g à une réaction potentielle) plus sa confiance augmente [31]. Tout en conservant la confiance bornée entre 0 et 1. Ainsi pour n confiances correspondantes de valeur moyenne Moy :

$$Confiance\,Globale = (1 - Moy) \times (1 - \frac{1}{n}) \times Moy + Moy \tag{4.2}$$

Cette fonction permet plusieurs propriétés. Ainsi, la confiance globale est toujours nulle (respectivement maximale et égale à 1) dans le cas où l'information est totalement fausse, i.e. $Moy = 0$ (respectivement totalement vraie avec $Moy = 1$). Les singletons ne sont pas modifiés. Autrement la confiance croît avec le nombre n des possibilités.

4.3.3.3 *Algorithme d'obtention de la confiance*

L'algorithme permettant le calcul de confiance est à nouveau séparé en deux : un point d'entrée (Fig. 65) et un processus principal (Fig. 66). Ce processus principal peut être accéléré si les règles utilisées ont des causes indépendantes (Fig. 67). Le processus principal est appelé directement pour la requête ou successivement pour

30. La moyenne a d'abord été considérée. Mais il faut une non linéarité pour discriminer des cas utilisant la notion de contraire qui créent un ensemble de chemins implicites et linéaires. La combinaison de ces chemins donnent alors toujours la même valeur à des cas que l'utilisateur est capable de différencier. Le changement des diverses fonctions est prévu dans le moteur.

31. De plus, une règle ayant beaucoup de causes offre en général beaucoup de chemins menant aux effets. Ainsi si le produit des confiances des nombreuses causes diminue la confiance élémentaire des effets, le nombre de chemins augmente la valeur globale. Ainsi les différentes règles ont un comportement relativement équilibré malgré le champs des possibles.

chaque terme élémentaire dans le cas d'une liste et la confiance est alors le produit de la confiance en chacun (Fig. 65).

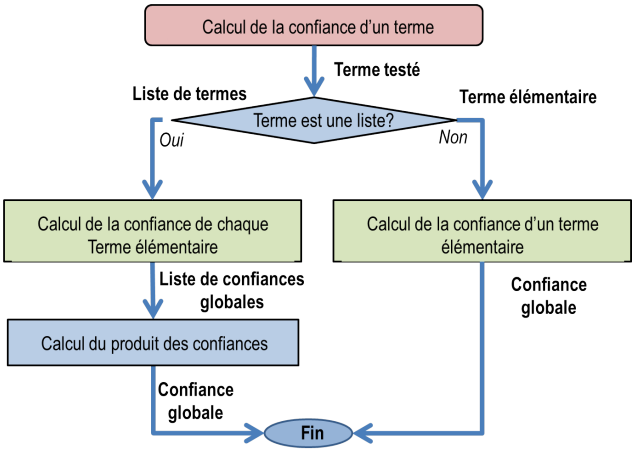


FIGURE 65: Algorithme pour calcul de la confiance

Pour chaque terme élémentaire, le processus commence par vérifier que l'on a affaire à un CG (Fig. 66). Dans le cas contraire, il fait appel à l'interpréteur PCG et renvoie la confiance maximale 1 si l'unification est possible et 0 dans le cas contraire. Pour un CG, l'algorithme s'intéresse à l'ensemble des chemins menant à une possibilité de confiance. Contrairement à l'algorithme de vérité, on s'intéresse à la fois aux assertions, directement présentes dans la base, et aux confiances issues de nos règles.

Pour les assertions, différents formats d'expression de confiance initiale sont définis avec leurs propriétés (section 4.3.3.2). Lorsque l'ensemble des formats ont été parcourus, le résultat est une liste de confiances liées à ces possibilités. Une étape de terminaison est alors en général nécessaire pour traiter les effets secondaires ou des redondances non désirées dus à la méthode de recherche de l'ensemble de ces formats.

Sont également recherchées en même temps les confiances issues de règles. Celles-ci sont plus complexes à obtenir. Pour chaque règle pouvant correspondre, l'ensemble des chemins amenant à des causes vraies et différentes pour l'effet recherché est obtenue. Ces chemins sont stockés comme autant de règles virtuelles. Cette étape est nécessaire pour gérer la dépendance entre les causes. Des sauvegardes intermédiaires sont effectuées à cette étape pour accélérer l'exécution. Puis, l'algorithme de confiance dans son ensemble est appelé avec comme entrées successives les termes élémentaires de ces causes unifiées. Leur produit pondéré par la confiance de la règle est alors stocké. Lorsque l'ensemble des règles correspondantes ont été parcourues, une liste des confiances issues des règles est obtenue.

La fusion finale est effectuée sur la concaténation des listes de confiance issues des assertions et des règles. Cette liste correspond bien aux différentes confiances d'une même situation, la requête initiale. Contrairement aux listes précédemment rencontrées dans l'algorithme qui étaient des confiances de descriptions différentes (le cas d'une liste comme requête ou représentant des causes). La fonction combinant les informations concernant la même situation a été présentée section 4.3.3.2.

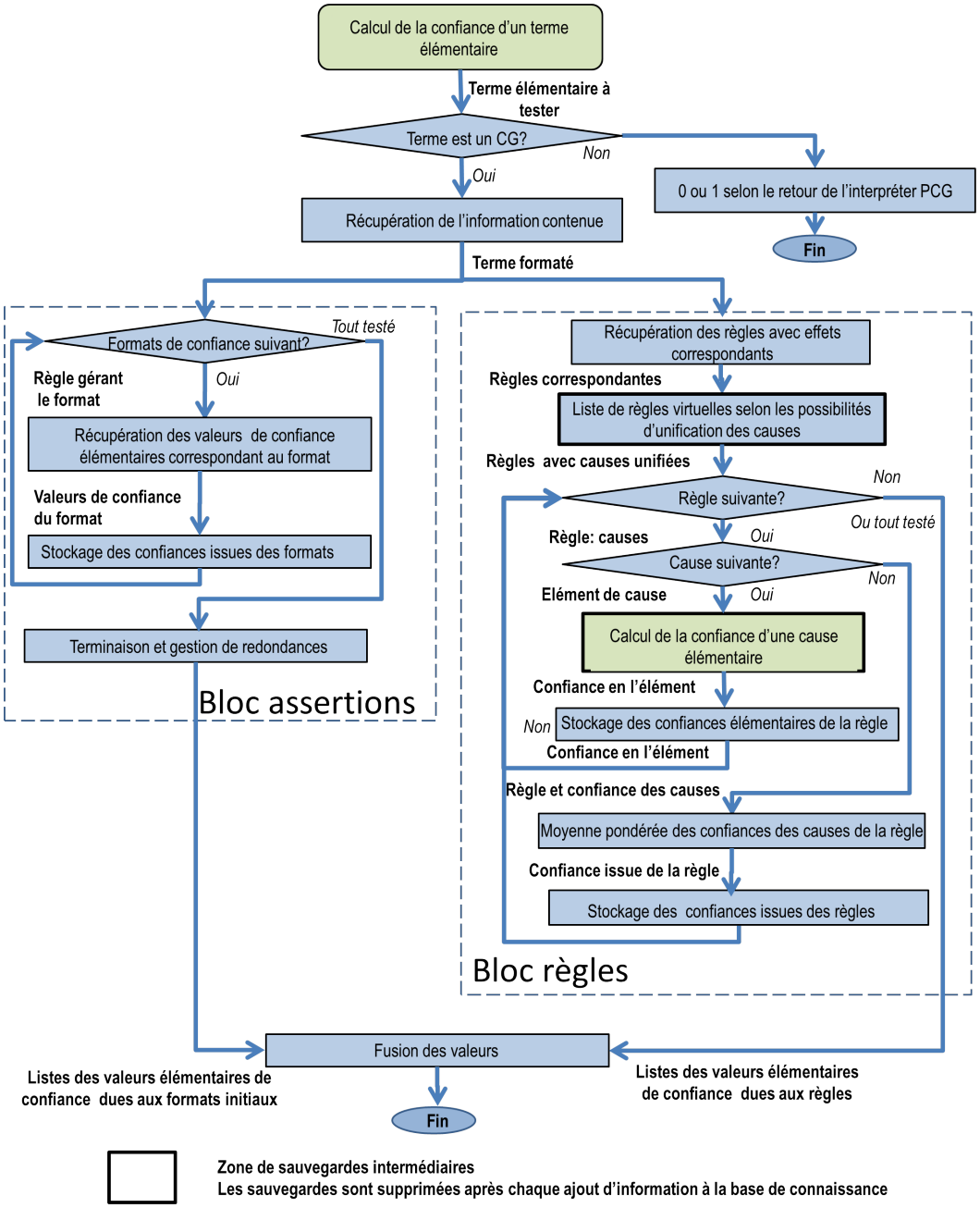


FIGURE 66: Algorithme pour le calcul de la confiance d'un terme élémentaire

Une version permet d'obtenir plus rapidement la confiance des règles dont les causes sont indépendantes (Fig. 67). Pour chaque règle pouvant correspondre, après le test de vérité des pré-conditions, le calcul de la confiance des causes est effectué. Celui-ci appelle l'algorithme dans son ensemble avec comme entrées successives les termes élémentaires des causes, non unifiés au préalable et donc sans appeler ici l'algorithme de vérité (évitant le double parcours de la base de connaissance). L'algorithme principal est arrêté dès qu'une des causes retourne une confiance nulle [32]. En effet, une confiance nulle équivaut à un terme faux parmi les causes, invalidant ainsi la règle. Dans le cas où ces confiances sont obtenues, et les post-conditions vérifiées, leur produit pondéré pour cette règle est stocké. Lorsque l'ensemble des

32. Ce n'est pas uniquement pour l'optimisation. La fonction de combinaison de confiance des listes peut être choisie et par exemple correspondre à la moyenne. Or la moyenne n'aurait pas été forcément nulle alors que la règle est fausse.

règles correspondantes ont été parcourues, une liste des confiances issues des règles est ainsi obtenue.

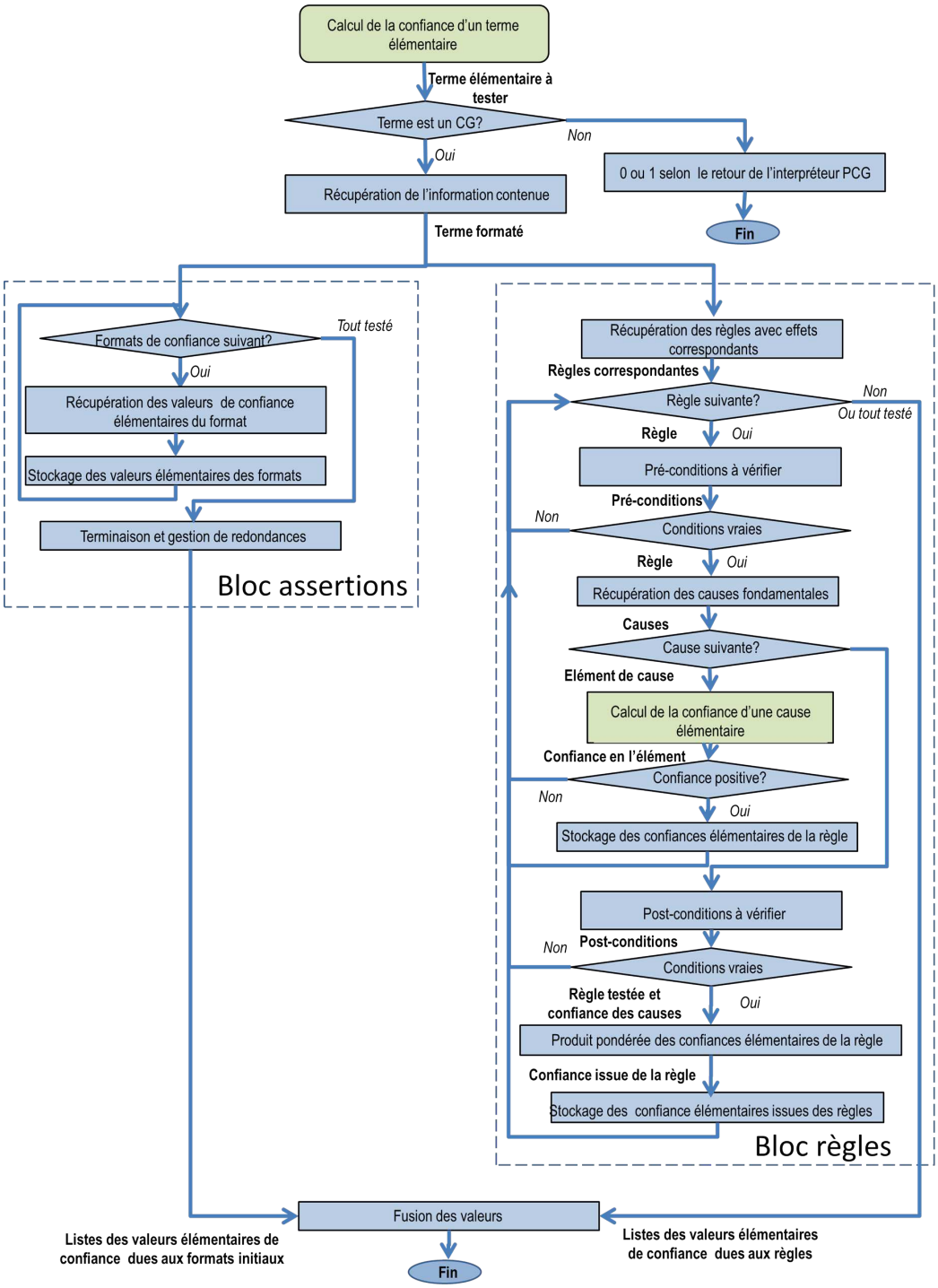


FIGURE 67: Algorithme pour le calcul de la confiance d'un terme élémentaire avec règles aux causes indépendantes

4.3.3.4 *Discussion sur les algorithmes de confiance et les optimisations*

L'obtention d'une version plus rapide pour les règles à causes indépendantes n'était pas prévue. En réalité cet algorithme a été utilisé en toute généralité tout en produisant les bons résultats pour les deux premières applications. En effet,

en pratique toutes nos causes étaient indépendantes. Cela n'est pas si surprenant car l'indépendance qui nous intéresse est celle après l'unification des effets. Or souvent, la dépendance entre les causes est liée à des variables présentes dans les effets. Celles-ci étaient alors naturellement unifiées par la requête initiale (issue du processus de réaction ou du concepteur) afin de savoir si la règle peut correspondre...

Or, le calcul de confiance est le processus le plus complexe. Et il s'est alors à nouveau complexifié. C'est pour cela qu'il est la première cible d'optimisation. Les optimisations sont toujours en cours de développement et sont présentées dans les perspectives de cette thèse. Néanmoins parmi les possibilités d'optimisations typiques de Prolog [15] se trouvent les sauvegardes intermédiaires. Celles-ci sont alors presque obligatoires et simultanément faciles à mettre en oeuvre dans la version générale de l'algorithme. En effet celui-ci parcourt l'ensemble des unifications possibles selon la base de connaissance. Il est alors possible de sauvegarder les résultats précis sans risque d'erreurs. Actuellement les valeurs de confiances et les règles virtuelles issues de l'algorithme général sont sauvegardées et supprimées lors d'une modification de base de connaissance. Cela permet finalement peu à peu d'analyser une base de connaissance statique et permet alors de répondre de plus en plus rapidement à n'importe quelle requête, en se basant sur les étapes précédentes.

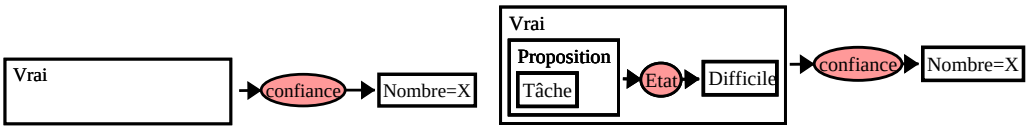
Cependant, cette sauvegarde ne peux être directement implémentée pour l'algorithme précédent. En effet, des possibilités d'erreurs existent lors de sauvegardes de CGs avec variables non unifiées. En effet, la sauvegarde est efficace car elle court-circuite le raisonnement et ses calculs. Or une requête spécifique peut s'unifier avec de nombreux CGs possédant une variable libre. Ainsi, pour une sauvegarde correspondant à une valeur de confiance, celle récupérée ne sera souvent pas la bonne (elle sera typiquement plus importante car correspondant à un CG plus général). La raison qui rend les sauvegardes d'optimisations difficiles dans notre cas est l'ouverture du moteur. Chaque étape peut être modifiée ou combinée de nombreuses façons et peu de sauvegardes sont compatibles avec l'ensemble des usages possibles.

4.3.4 *Conclusion sur les processus de vérité et de confiance*

Les processus de vérité et de confiance, ainsi que l'influence du temps, ont été prévus pour être génériques. Ils sont ainsi compatibles avec les règles et faits PCG. De plus, s'ils introduisent des structures spécifiques, les processus sont valides quelle que soit la situation contenue dans ces structures. Ainsi la validité et l'évolution temporelle d'une situation quelconque, incluant les informations de temps présentées, sont gérées de manière transparente. Le plus intéressant est la gestion de la confiance : quelle que soit la requête, les informations seront combinées pour quantifier le degré de vérité associé. Il devient ainsi possible de quantifier une situation quelconque selon sa description sémantique. Ce CG descriptif est à insérer dans le concept *vrai* via l'expression de la structure [33] représentée Fig. 68a, pour obtenir le nombre associé à sa *confiance* globale. Ainsi obtenir la confiance de l'existence d'une difficulté de la tâche (e.g. Fig. 68b) tient compte de l'ensemble

33. Il devient éventuellement possible d'imposer la confiance globale d'une situation en ajoutant cette structure avec un nombre donné à la base de connaissance. Seul le dernier ajout est pris en compte.

des descriptions sémantiques compatibles et des chemins permettant d'établir leur confiance. Incluant par exemple, une difficulté de l'interaction due à un mouvement utilisateur anormal et une difficulté de visualisation de la tâche due à une cible cachée. Cela permet également de pouvoir quantifier et comparer des situations à la sémantique différente. Un processus générique de réaction peut donc être obtenu, tout en bénéficiant d'un raisonnement sémantique.



(a) Format d'obtention de la confiance globale (b) Exemple de calcul de confiance globale

FIGURE 68: Exemples de CGs permettant d'obtenir une confiance globale

4.4 PROCESSUS GÉNÉRIQUE DE RÉACTION

Le processus de réaction vise à obtenir automatiquement le déclenchement des assistances adéquates à la situation. Il permet ainsi sans expliciter de demande de réunir les réactions potentielles, en se basant sur le contexte et notamment les outils disponibles, et d'en sélectionner les meilleures. Le contexte est décrit de manière sémantique et le processus doit donc être capable de comparer des réactions au sens différent. Pour cela la notion de confiance en chaque réaction est utilisée et la notion d'impact introduite. Les commandes externes des réactions effectives visant l'application peuvent alors être élaborées.

4.4.1 *Types de réactions*

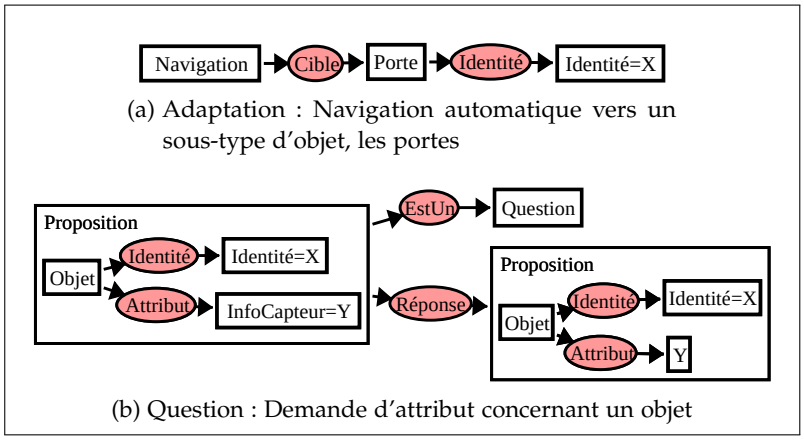
Les réactions sont, pour le moteur, des CGs qui peuvent déclencher des *outils*. Les adaptations (e.g Fig. 69a) ne nécessitent aucune structure spécifique et dépendent uniquement de la description d'*usages* d'*effecteurs*. Il est possible de déclarer explicitement des adaptations potentielles dans une *proposition* reliée par la relation *EstUn* au concept d'*adaptation*, afin de gérer certains cas complexes.

En revanche, les questions (e.g Fig. 69b) présentent des concepts spécifiques pour permettre au moteur de les gérer. La question est donc la description d'une *proposition* liée au concept de *question* par la relation *estUn*. Les informations obtenues par cette question correspondront aux variables libres contenues dans les descriptions des concepts *InfoCapteur*. Ces variables doivent être mises en valeur car d'autres variables libres classiques peuvent exister. Les variables classiques peuvent être unifiées au cours du processus permettant d'élaborer la question. Les variables des occurrences *InfoCapteur* ne peuvent, et ne doivent pas, être unifiées par le moteur. En effet, elles servent de réceptacles aux informations retournées par l'application visée. Enfin, un concept supplémentaire peut être ajouté à la structure des questions pour introduire la suite élémentaire du raisonnement à effectuer au retour de l'information. Cette *réponse* est ensuite ajoutée à la base de connaissance. Cela permet d'obtenir un raisonnement différent en fonction du contexte[34] où la

34. Par exemple, un même outil, permettant de récupérer une information donnée, peut être ainsi utilisé différemment selon la description de la *réponse* à la question posée.

question a été posée.

La Fig. 69a est une situation qui décrit ainsi la navigation vers un objet. Cette situation peut devenir une adaptation dès lors qu'un outil permet de l'implémenter (e.g. Fig. 73a). Fig. 69b est une question qui permet d'obtenir les attributs d'un objet donné. Le concept *InfoCapteur* permet notamment de spécifier que c'est l'attribut et non l'identité de l'objet qui est ici demandé. Enfin, la gestion de la réponse à la question est ici explicitée mais est identique à celle par défaut et permet d'ajouter l'information non modifiée à la base de connaissance.



(a) Adaptation : Navigation automatique vers un sous-type d'objet, les portes

(b) Question : Demande d'attribut concernant un objet

FIGURE 69: Exemples de réactions

4.4.2 *Notion d'impact et comparaison des décisions*

Le processus de décision doit pouvoir comparer des réactions potentielles qui possèdent des sémantiques très différentes. La *confiance* est un concept général qui peut être agrégé pour toutes informations, notamment les réactions. Une mesure du degré de vérité des réactions selon la situation est donc possible. Cependant et selon sa sémantique, l'ordre de grandeur du degré de vérité nécessaire pour décider de l'application d'une réaction est variable. Ce seuil variable peut être associé à un concept, l'*impact*, représentant la mesure du degré d'effet sur l'utilisateur de la décision. Ce concept pourra être ainsi associé naturellement à la *confiance* avant la prise effective de décision : plus une décision a d'impact, plus il faut être confiant en sa vérité actuelle avant de l'appliquer.

Il est difficile de décider de la valeur de l'*impact* selon la situation décrite en toute généralité. Par contre, les décisions étant appliquées par les *outils*, la valeur initiale peut être facilement introduite par la description sémantique de ceux-ci. Il est alors aisé de permettre la modification de l'*impact* initial, en fonction de l'*usage* de l'*outil* et du contexte, via la définition de *cas*. Une réaction, correspondant à l'*usage* d'un *cas* actuellement vrai, subit la *modification* de son *impact* indiquée par celui-ci . Par exemple, un cas augmente l'impact d'une activation répétée. Cela permet de stabiliser le moteur pour atténuer les possibilités de boucle d'activations/désactivations.

Enfin, même pour des réactions adéquates, appliquer indistinctement l'ensemble des réactions potentielles simultanément risque de perturber l'utilisateur plus que de l'aider. Limiter arbitrairement le nombre de décisions est une possibilité, mais elle ne tient pas compte de la sémantique des réactions. C'est pourquoi la définition

de la notion d'impact est doublement intéressante. En limitant la quantité d'impact total utilisable, la saturation de l'attention de l'utilisateur est évitée.

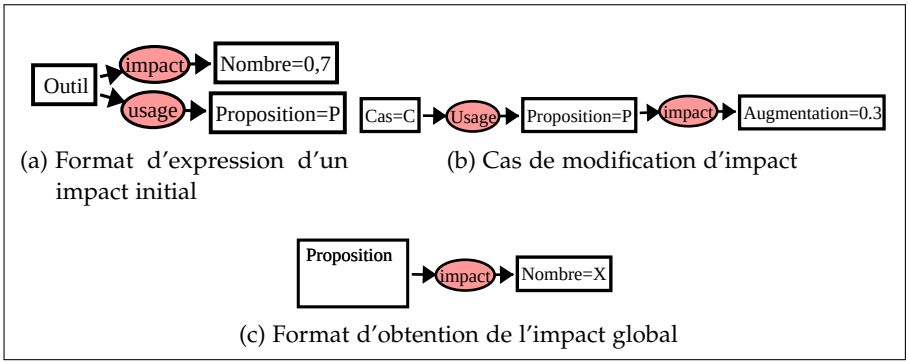
De plus, cet impact total permet potentiellement d'ajouter des règles pour élaborer cette limite plus finement en fonction de la sémantique (par exemple permettre une certaine quantité pour chaque modalité afin d'éviter la saturation des différents canaux proprioceptifs). L'impact admissible est égal à l'impact indiqué comme supporté par l'utilisateur (un premier pas dans l'établissement d'un profil) auquel on soustrait l'impact déjà actuellement utilisé (correspondant aux réactions encore actives voir section 4.5.2.1).

Ainsi le calcul de l'*impact* est géré pour les réactions. Chaque *outil* fournit un *impact* initial (e.g. structure simplifiée Fig. 70a) qui peut être modifié selon certains *cas* dont l'*usage* correspond (e.g. structure Fig. 70b). Les *impacts* initiaux égaux à 0 (sans aucun impact) ou égaux à 1 (avec l'impact maximum) ne sont pas modifiés. Autrement, pour chaque *cas* applicable, l'*impact* est modifié suivant un poids (W, égal à 25% si le poids n'est pas précisé dans le CG), tout en restant borné entre 0 et 1.

$$impact(n) = impact(n-1) - W \times impact(n-1) \text{ pour un } impact\ supérieur \tag{4.3}$$

$$impact(n) = impact(n-1) + W \times (1 - impact(n-1)) \text{ pour un } impact\ inférieur \tag{4.4}$$

Ainsi de plus petits pas sont effectués pour les valeurs d'impacts extrêmes. Cela permet de garder l'impact borné. Cela a également pour effet de permettre aux adaptations ayant déjà beaucoup d'impact de rester atteignables ; ainsi que de garder un seuil effectif pour les décisions déjà très faciles à atteindre. Pour obtenir l'impact global d'une réaction, la structure [35] Fig. 70c est utilisée où la réaction [36] envisagée est contenue dans la *proposition*.



(a) Format d'expression d'un impact initial

(b) Cas de modification d'impact

(c) Format d'obtention de l'impact global

FIGURE 70: Exemples de CGs exprimant des propriétés d'impact

35. Il devient aussi possible d'imposer l'impact global avec cette structure.

36. Toute description est possible mais s'il n'existe pas d'outil correspondant, l'impact global retourné est nul.

4.4.3 *Algorithmes de réaction*

Concernant le processus de réaction, il est possible de choisir que les questions soient prioritaires (première étape Fig. 71 ou 72). Dans ce cas dès qu'une question a été posée, il n'est plus possible de déclencher des réactions tant que la réponse à la question n'a pas été reçue. L'idée est d'attendre la construction complète du contexte afin d'éviter des décisions potentiellement hâtives. Il est en effet possible de bloquer des décisions plus pertinentes à l'issue du nouveau contexte. Par exemple, en ayant déjà dépensé l'impact disponible dans des adaptations plus génériques et peut-être mal adaptées au vu des réponses aux questions. Cependant cette option pose problème, dans le cas où le moteur n'est pas assez rapide par rapport à la fréquence d'évolution du contexte, et est désactivée par défaut. Dans ce dernier cas, il y a en effet toujours de nouvelles questions à poser, sans avoir le temps d'effectuer de cycle entre les réponses pour obtenir des adaptations.

D'ailleurs les questions ont une autre spécificité. En effet, pour une *question* effective à l'issue du processus, une commande avec une ou des variables libres est envoyée au capteur. Celui-ci renvoi par la suite le même genre de commande en retour. Ce retour est alors utilisé pour terminer l'unification de la *réponse* décrite par le capteur. Cela permet de reprendre en partie le raisonnement qui avait abouti à cette question afin d'effectuer les conclusions appropriées. Le processus de décision doit donc sauvegarder les questions effectivement posées (gestion des questions Fig. 71 ou 72).

Ainsi, l'algorithme de réaction commence par vérifier si les questions sont prioritaires. Une décision est considérée par le processus si elle est à la fois vraie au vue de la situation et applicable via un *outil*. Le processus cherche ensuite à comparer la confiance et l'impact de chaque réaction, qui sont alors calculés. Par la suite, seules les réactions les plus pertinentes, avec un rapport de confiance sur impact suffisant, sont conservées à l'étape de classification.

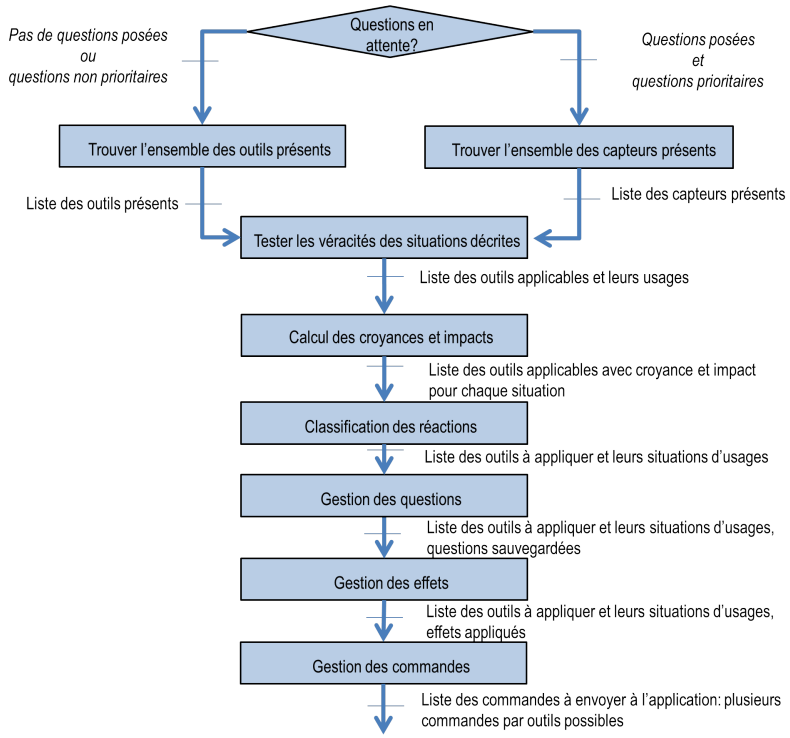


FIGURE 71: Algorithme général de réactions : approche globale

Plusieurs processus de réactions sont disponibles sur cette base. Le principal utilisé considère l'ensemble des décisions potentielles simultanément Fig. 71. Ces décisions sont ensuite optimisées afin de maximiser la somme de leur confiance en gardant leur somme d'impacts compatible avec l'impact disponible. C'est une optimisation de type sac à dos qui se produit également à l'étape de classification des décisions. Il est possible au besoin de modifier la fonction de classification. Par exemple de tenter d'accélérer cette étape en sélectionnant les décisions directement selon leur ratio de confiance sur impact[37].

Un algorithme approché peut être utilisé (Fig. 72). Il sélectionne les décisions suffisamment pertinentes à mesure qu'elles sont trouvées. Bien que sensible à l'ordre des informations dans la base, il permet une approximation plus rapide à exécuter dans certains cas. En effet, le processus peut être arrêté dès que l'impact restant est suffisamment bas.

37. Ce qui n'est actuellement pas intéressant au vue du faible nombre de décisions simultanément applicables au final.

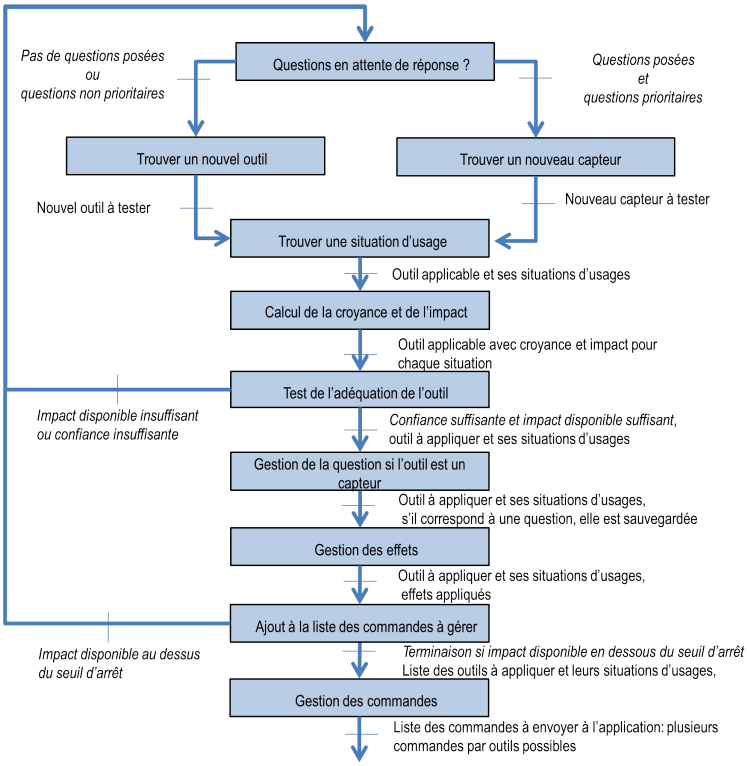


FIGURE 72: Algorithme approché de réaction : approche séquentielle

A l'issue de la sélection des décisions à déclencher (ou une à une dans le cas de l'algorithme approché) viennent les étapes permettant de gérer les effets des réactions et de déterminer les commandes à envoyer à l'application. Ces étapes correspondent à la réification des décisions et sont présentées ci-après.

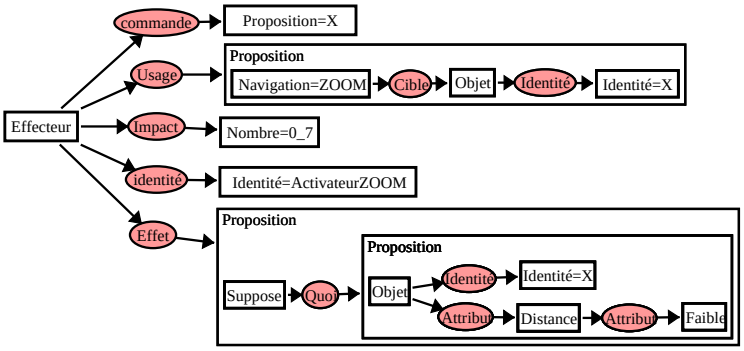
4.4.4 *Réification des décisions et communication*

Les réactions effectives décidées par le moteur sont ensuite réifiées à la fin du processus de décision. Cela correspond pour l'application, à l'envoi de messages grâce aux *commandes* spécifiques des *outils* déclenchés. Ainsi la description d'un outil (e.g. Fig. 73) doit contenir son *usage*, l'*identité* de l'outil ainsi que la *commande* à envoyer. De plus, à moins d'avoir un impact nul, les outils doivent fournir dans leur description l'information d'*impact* initial. Mais la réification correpond aussi pour le moteur au déclenchement d'*effets* éventuels liés à chaque outils. Ainsi, les outils peuvent avoir des *effets*, explicitement décrits (Fig. 73a) ou implicites liés à leur *usage* (Fig. 75b section suivante).

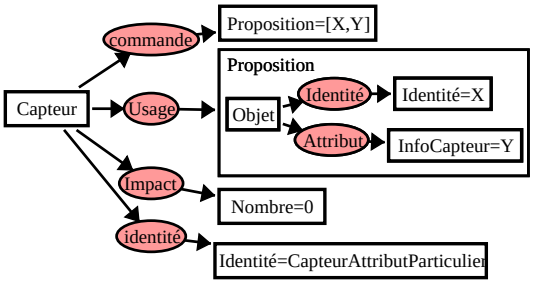
Par exemple, Fig. 73a est un *effecteur* permettant d'effectuer un zoom sur un objet. La décision effective le déclenchant été obtenue par une unification réussie de son *usage* (par exemple à l'aide du CG Fig. 69a). La description de cet effecteur a un impact assez élevé car le point de vue de l'utilisateur sera automatiquement modifié suite à son application. De plus, il aura également pour effet, explicitement décrit, de supposer que l'objet est désormais proche. Cela peut notamment être intéressant [38] si l'application ne possède pas de capteur pouvant obtenir par ailleurs cette information. Un autre exemple, Fig. 73b est un *capteur* permettant d'obtenir les

38. Cela peut être également utilisé comme base pour le planning.

attributs d'un objet donné. Il peut être déclenché également dès lors que son *usage* a été unifié (via un CG statuant sur une question donc, par exemple Fig. 69b). Cela est totalement transparent pour l'utilisateur et a donc un impact nul (par défaut, lorsque non explicitement décrit).



(a) Effecteur pour se rapprocher d'un objet



(b) Capteur pour obtenir les attributs d'un objet

FIGURE 73: Exemples d'outils

La communication est gérée grâce à la description de la réaction effective et de l'*outil* correspondant. Ainsi, l'*identité* de l'outil permet au protocole de communication de contacter l'entité correspondante dans l'application. La description de la *commande* via une liste permet de ne pas supposer que les outils soient en mesure de comprendre la sémantique des CGs. Ils peuvent ainsi décrire eux-même via n'importe quelles entités de PCG ce qu'ils doivent recevoir comme information. Par exemple une liste de mots-clefs, de chiffre etc. Il est ainsi possible d'obtenir une liste de mot-clefs automatiquement en associant des variables de la commande et de la description de l'usage.

Le processus gérant le moteur et sa communication peut ensuite envoyer la commande unifiée sur le canal portant le nom de l'outil. C'est une implémentation naturelle du protocole OSC mais qui nécessite que chaque outil soit capable d'avoir un écouteur OSC. Une autre version de la communication permet d'ajouter un séparateur pour accoler la liste des couples contenant identité et commande. Ce qui permet ensuite à l'application de répartir cet unique message comme elle l'entend, à partir d'un unique écouteur centralisé.

D'ailleurs c'est la communication qui fait la différence pour les méta-outils. Ainsi la commande des méta-effecteurs n'est pas envoyée. Si elle possède une description, elle est au contraire évaluée par le moteur. Il est également possible de se

servir des *effets* des outils, qui peuvent être ajoutés notamment aux méta-outils. Les méta-capteurs ne sont pas encore utilisés. Il suffirait de choisir comme *identité* de l'outil celle correspondant au canal OSC écouté par le module externe. Les outils externes ne sont alors pas traités différemment. En revanche dans le cas d'une communication centralisée par l'application, il faut ajouter la gestion d'un concept de cible de l'outil, pouvant viser l'application principale ou un autre module externe.

La gestion des outils ne se fait donc pas sans considérer leur sémantique, notamment leur type. Et la description de leur *usage* est fondamentale pour décider de leur application. Mais il est possible également de définir des effets secondaires basés sur ces descriptions. Un effet secondaire spécifique est la gestion des *cas* modifiant l'*impact* des réactions aux *usages* correspondant (au moment du calcul d'impact). Un autre est le déclenchement d'*effets automatiques* pour chaque décision effective dont, à nouveau, l'*usage* correspond (comme pour l'activation section 4.5.2.1).

Enfin certaines sécurités[39] sont implémentées pour gérer les grandes variétés d'unifications possibles. Ainsi si un *outil* est un *effecteur*, il ne doit pas avoir de variables libres après unification pour être valide. Cela est très pratique notamment pour éviter de déclencher des effecteurs qui n'auraient pas prévu de cas par défaut pour une variable non unifiée. C'est le cas le plus courant puisque, l'outil au sein de l'application est globalement indépendant de la couche logique. Au contraire, un *capteur* doit avoir au moins une variable libre dans sa commande, qui est utilisée comme réceptacle de la réponse de l'application.

4.4.5 *Paramétrages du moteur*

A l'aide des processus présentés, des demandes spécifiques et une demande générique de réaction peuvent être résolues par le moteur, qui gère automatiquement le temps et un degré de vérité. L'usage du moteur se fait en pratique après l'ajout de règles et de faits constituant la base de connaissance initiale. Ces modifications correspondent au fonctionnement classique du moteur. Mais ces règles de comportements peuvent dépasser le cadre d'une application particulière et introduire la gestion de nouveaux concepts. Ces extensions globalement réutilisables sont présentées à la section suivante. Les règles de comportements spécifiques, visant plus particulièrement l'assistance à l'interaction 3D, sont elles présentées au chapitre suivant.

Mais il est encore possible de paramétrer le moteur autrement, en profitant des différents points de configuration introduits. Ces configurations peuvent être décrites pas des CGs et affectent les différents processus du moteur. La modification de ces paramètres à l'aide d'un méta-effecteur est également une des possibilités de méta-adaptations.

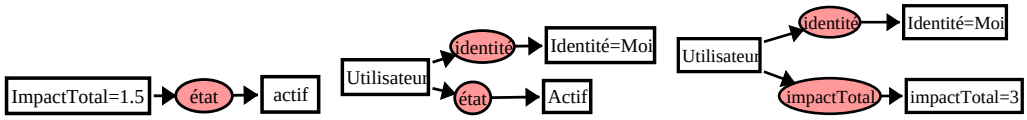
4.4.5.1 *Modification de l'algorithme de réactions*

Le processus de réaction possède ainsi différents paramètres aisément modifiables :

39. Si l'on veut s'abstraire de ces tests, il suffit de décrire l'*outil* via ce même concept général, sans en préciser le sous-type.

- le type d'algorithme (pour règles indépendantes ou général) ;
- rendre ou non les questions prioritaires ;
- le ratio de confiance sur impact minimal nécessaire ;
- l'impact total admissible ;
- le mode de raisonnement (algorithme approché ou général)
- les modes de classification pour les décisions des algorithmes (problème de type sac à dos, maximum de pertinence, etc.) ;
- l'impact minimal pour continuer l'algorithme approché ;
- la fonction qui associe une confiance globale à une liste de confiances élémentaires d'une même description (la fonction de fusion, une moyenne, etc.) ;
- la fonction qui associe une confiance globale à une liste de confiances de descriptions a priori différentes (le produit etc.).

Ces modifications peuvent être paramétrées par des CGs. Le moteur peut reconnaitre directement la définition [40] de l'impact total admissible (Fig. 74a). Cette définition, comme celle de tout CG peut être modifiée ou issue de règles. Ainsi, cette information peut être issue d'une règle liant l'utilisateur actif (Fig. 74b) et la définition d'éléments de profil (Fig. 74c). L'utilisateur courant peut donc choisir d'obtenir plus d'adaptations effectives que celles définies par défaut.



(a) Définition de l'impact total admissible (b) Élément de session active (c) Élément de profil utilisateur

FIGURE 74: Exemples de paramétrisation de l'impact total par des CGs

4.4.5.2 *Modification des formats de vérité et de confiance*

Il est possible d'altérer les processus du méta-interpréteur via les différents formats permettant de traiter l'information. Il est ainsi possible de modifier, d'ajouter ou de supprimer les formats permettant de déterminer ce qu'est :

- une assertion de la base de connaissance. Il est facile d'ajouter un sous-concept du *fait* qui obtiendra les mêmes propriétés (sa description sera considérée vraie également). Mais n'importe quelle interprétation de CG afin d'en extraire une information vraie peut être définie. Elle sera gérée à tous les niveaux par le moteur (déclenchement de règle, confiance unitaire, etc.).
- un information de confiance initiale. De même, n'importe quelle interprétation de CG afin d'en extraire un nombre peut être définie.
- une règle liant des informations. L'interprétation de CGs qui fournie une liste de causes, d'effets, de pré-conditions et de post-conditions pour chacun et un nombre pour la confiance de la règle.
- une adaptation. C'est une modification d'une sous-catégorie de la définition de vérité. Cela permet de définir une structure supplémentaire sous laquelle le moteur est autorisé à rechercher les adaptations.

40. La définition peut être écrite de manière plus simple avec l'attribut total associé à l'impact. Mais pour éviter une ambiguïté pour les paramètres du moteur, sans traitement des entrées utilisateur, des concepts spécifiques liés au moteurs sont introduits.

– une question. Une définition de structure comme pour les adaptations mais plus complexe. Le format doit fournir également un CG qui sera unifié au retour d'information via un deuxième appel à la structure.

La définition de formats peut provoquer des usages multiples d'une même information. Ceux-ci non-voulus ou non-maitrisés peuvent être gênants (annexe A). Cela peut fausser potentiellement l'ordre de grandeur de la confiance obtenue. La modification n'est donc ici pas toujours évidente car il faut parfois gérer la priorité des formats. Certaines modifications des notions initialement proposées ont déjà été apportées et sont présentées dans la section suivante, avec notamment une gestion différente de la notion de faux.

4.5 EXTENSION DES CONCEPTS DISPONIBLES POUR LE RAISONNEMENT

Certaines règles issues de besoins rencontrés par nos applications ont une portée plus générale. Ainsi, sont présentés ici les concepts et relations introduits qui sont aisément réutilisables pour l'expression de situations. Certains touchent plus particulièrement les notions de vérité et de confiance même (section 4.5.1). D'autres sont de nouveaux concepts avec leurs propriétés. Ainsi des concepts d'états *actifs* et *disponibles* possèdent une gestion particulière (section 4.5.2). D'autres permettent des interprétations multiples selon la structure du graphe qui les utilise (section 4.5.3). Finalement les derniers introduits combinent des ensembles d'informations, comme le *choix* ou l'*agrégation* (section 4.5.4).

4.5.1 *Extensions de la notion de vérité et de confiance*

L'expression de vérité est parfois insuffisante. Pouvoir spécifier la négation d'une situation ou des termes contraires est ainsi utile. Il faut alors calculer leur confiance en adéquation avec leur sens. Enfin des notions qui combinent les informations et étendent également en pratique vérité et confiance sont présentées section 4.5.4.

4.5.1.1 *Notion de fausseté*

Ne pas posséder le concept de *faux* empêche de définir une cause comme étant la négation d'une situation particulière. Ce qui est par exemple nécessaire pour écrire qu'un objet n'est pas un intérêt. Cette situation de départ permet notamment la suppression d'une modalité précédemment introduite. Le concept de *faux* doit donc être géré.

Le moteur prend en compte la notion de vrai et de confiance. Le faux est défini par défaut comme l'absence de vrai. Mais il faut pouvoir l'exprimer directement à l'aide d'un CG sous peine de limiter les possibilités. Dès lors, la gestion de la confiance du *faux* doit être spécifique pour rester cohérent. Notamment il faut lier cette confiance à celle de l'expression correspondant à la vérité de cette même situation. Sans gestion spécifique une situation fausse ne peut être que binaire, ce qui gênerait la cohérence et la propagation dans les règles de la confiance issue de ce concept. Ainsi la gestion du *faux* introduit sa vérité mais également sa confiance.

La confiance d'une proposition associée à la description[41] du concept *faux* est égale à 1 moins la confiance en la vérité de cette même situation.

4.5.1.2 *Notion de contraire*

Cependant gérer le concept de *faux* n'est pas équivalent à introduire un concept spécifique pour exprimer un *contraire*. Notamment lorsque un raisonnement qualitatif est sous-entendu. Par exemple, sans une gestion des *contraires*, exprimer la situation où il est faux qu'un objet soit plutôt loin, n'est pas équivalent à statuer sur le fait qu'un objet soit plutôt proche. Car sans information, le *faux* correspond à une gestion de l'absence et la confiance de la situation exprimée comme fausse est injustement favorisée. En effet si le capteur n'est pas présent et n'apporte donc pas l'information d'attribut de distance, la confiance qu'il soit faux qu'un objet soit loin est maximale.

Lorsque ce n'est pas l'absence mais la nuance qui est désirée, il faut donc pouvoir exprimer une situation et son contraire. Les informations doivent alors être reliées entre-elles via la définition de notions contraires. En effet une information sur l'une ou l'autre doit avoir une effet sur son opposé. En revanche, l'absence d'information ne doit favoriser ni l'une ni l'autre des notions. Ainsi la confiance d'une structure définie comme ayant un contraire (e.g. associé un concept et un attribut possédant un contraire) vérifie au préalable[42] l'existence de son opposé. Si une information contraire est disponible, la confiance de ce chemin supplémentaire est établie comme pour le *faux*. Si aucune information n'est disponible, les deux sont alors *faux*.

4.5.2 *Gestion d'états et localisation dans l'application*

De nombreux états peuvent être utilisés comme étapes du raisonnement d'assistance, parfois propres à l'application (section 5.4). Néanmoins certains comme les *états actifs* et *disponibles* ont des effets supplémentaires. Ils permettent de manière similaire à un *fait* localisé dans le *temps* ou à un *objet* localisé dans l'*espace*, de situer l'information vis-à-vis de l'application.

4.5.2.1 *Notion d'état actif et activation*

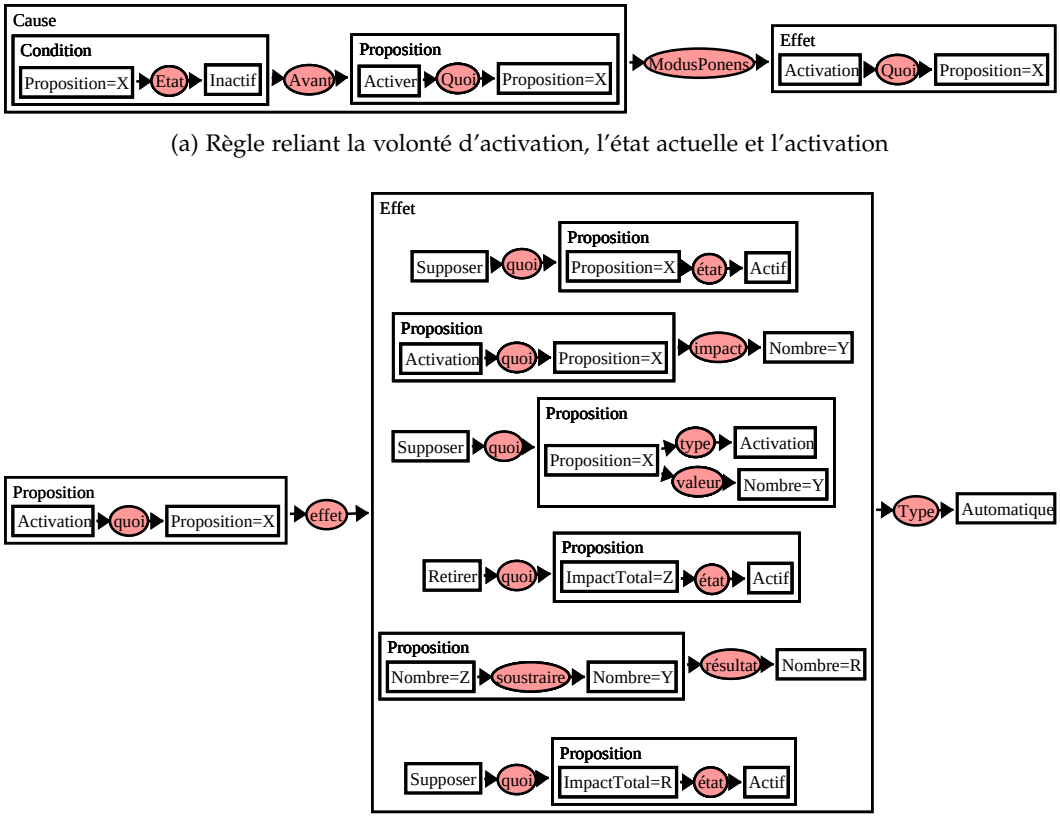
La notion d'*activation* a été introduite pour gérer les effecteurs dont l'effet avait une permanence. Par exemple, en décidant de valoriser un objet en lui attribuant la couleur rouge dans notre application, l'effet était permanent. Sans information retournée par l'application, le moteur continuait alors à prendre cette même décision, le contexte n'ayant pas changé. C'était à la fois incohérent pour l'utilisateur, ralentissait le processus de décisions et était également source d'erreurs pour l'effecteur au sein de l'application. L'*activation* est une notion nécessaire qui touche donc à la fois au temps et à la vérité.

41. En pratique, l'exécution est plus rapide lorsque une situation fausse est décrite dans une proposition menant au concept *faux* par la relation *faux*. PCG cherche en priorité les relations et les possibilités d'unification sont également moins nombreuses. Les deux expressions sont gérées, et si la version courte est illustrée par la suite, c'est la version rapide qui est implémentée.

42. Cette implémentation évite les boucles infinies en ajoutant un verrou pour l'utilisation de la règle. Le calcul de la confiance d'un attribut nécessite de vérifier celle de son opposé, qui lui aussi vérifierait à nouveau le premier sans cette sécurité (voir annexe A)

Il est ainsi très utile de pouvoir facilement exprimer qu'une adaptation reste active après avoir été déclenchée. Premièrement, avec cette information, le moteur peut gérer sa désactivation future. Ainsi le moteur gère les *activations* et *désactivations*. L'*activation* d'une réaction n'est possible que si la volonté d'*activer* une situation est vraie et que celle-ci n'est pas déjà *active* (Fig. 75a). De même, la volonté de *désactiver* ne peut mener à une *désactivation* effective que lorsque la réaction est déjà au préalable *active*.

Les *effets* d'une *activation* déclenchée via un *outil* sont gérés automatiquement. Ils permettent de supposer l'*état actif* de l'action activée. De plus si l'adaptation reste active, elle continue en fait d'impacter l'utilisateur. Ainsi l'*impact* de la réaction est retiré également de l'*impact total* disponible (Fig. 75b). La *désactivation* aura pour *effets automatiques* le retrait de l'*état actif* de la réaction ainsi que le rétablissement de l'*impact* précédemment consommé.



(a) Règle reliant la volonté d'activation, l'état actuelle et l'activation

(b) Effets automatiques d'un outil implémentant une activation

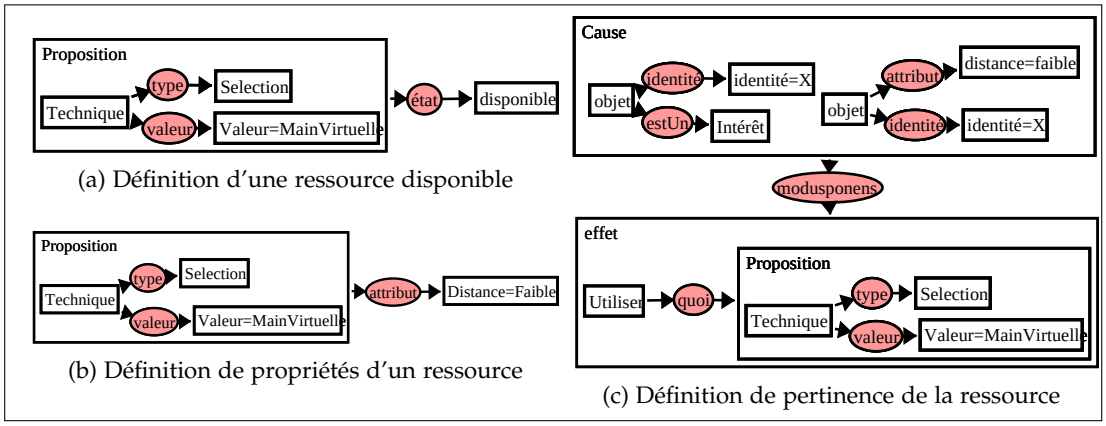
FIGURE 75: Règle et effets automatiques gérant l'activation

L'activation était initialement gérée par les outils au sein de l'application, puis directement par les effets explicites des outils au sein du moteur. Ainsi c'était l'application à travers ses messages ou sa description d'outil qui gérait l'état actif. Les contraintes sont alors plus grandes. De plus en ajoutant la notion d'impact dépensé, il est grandement préférable de gérer la notion d'activation en interne, comme présenté ci-dessus. Enfin, la description de l'effet automatique est longue car elle correspond à l'explicitation d'un processus. Il pourrait être géré de manière sémantique et haut niveau avec une notion avancée de ressources, simplifiant grandement l'expression Fig. 75b . Cette évolution pour la notion d'impact total est

ainsi considérée comme perspective. D'autant plus que la notion d'*état disponible* a été introduite pour gérer nos applications.

4.5.2.2 *Notion d'état disponible et de ressources*

La notion d'*état disponible* permet notamment d'exprimer la présence d'une ressource de l'application (e.g. Fig. 76a). On peut ainsi décrire le matériel sur lequel l'application fonctionne (type d'écran, affichage stéréoscopique ou non etc.) mais également des ressources virtuelles comme des techniques d'interaction ou des types de caméras. Un des types de ressources disponibles introduit, sans être ainsi nommés, sont les outils. Leur seule déclaration les rends implicitement disponibles [43]. Cependant on veut pouvoir exprimer des connaissances générales liées aux ressources dans la base de connaissance pour prévoir un raisonnement (e.g. les attributs naturels d'une technique d'interaction Fig. 76b, dont on peut spécifier une confiance selon l'adéquation). Selon la forme que prend notre expression, la ressource peut être directement citée dans la base sans pour autant affirmer que sa disponibilité est avérée pour l'application commandée. Il faut ainsi pouvoir expliciter ce qui est *disponible* (e.g. une technique d'interaction, Fig. 76a). Cette condition sera ainsi présente dans le raisonnement avant de considérer un usage particulier (e.g Fig. 76c).



(a) Définition d'une ressource disponible

(b) Définition de propriétés d'un ressource

(c) Définition de pertinence de la ressource

FIGURE 76: Illustrations de la notion de d'état disponible

Cette notion est actuellement uniquement une convention d'écriture pour des ressources non consommables (section 5.4). Mais il serait également intéressant d'introduire des *types* de ressources notamment consommables ou empruntables. Ainsi l'emprunt, le retour ou la consommation définitive d'une ressource sont alors autant d'actions à gérer automatiquement comme peut l'être l'activation.

4.5.3 *Règle d'interprétations de structures d'expression*

De nombreuses relations peuvent être interprétées différemment en fonction de la structure où elles apparaissent. Par exemple, la relation de *confiance* menant à un *nombre* permet d'exprimer :

- un degré de vérité élémentaire fixe lorsque reliée seule à un *fait* ;
- un degré de vérité élémentaire variable lorsque reliée à un *fait* qui possède également la relation *quand* ;

43. Les réactions ayant un état disponible sont alors celles correspondant aux usages des outils.

– un degré a priori de vérité élémentaire lorsque reliée à une *proposition* ;
– le degré de vérité lorsque reliée au concept de *vrai*.

Afin de limiter l'ontologie ainsi que de permettre une expression naturelle, de nombreuses relations peuvent être ainsi interprétées différemment en fonction de la structure où elles sont utilisées. Ainsi des relation intelligentes, dépendantes du contexte d'usage, sont définies.

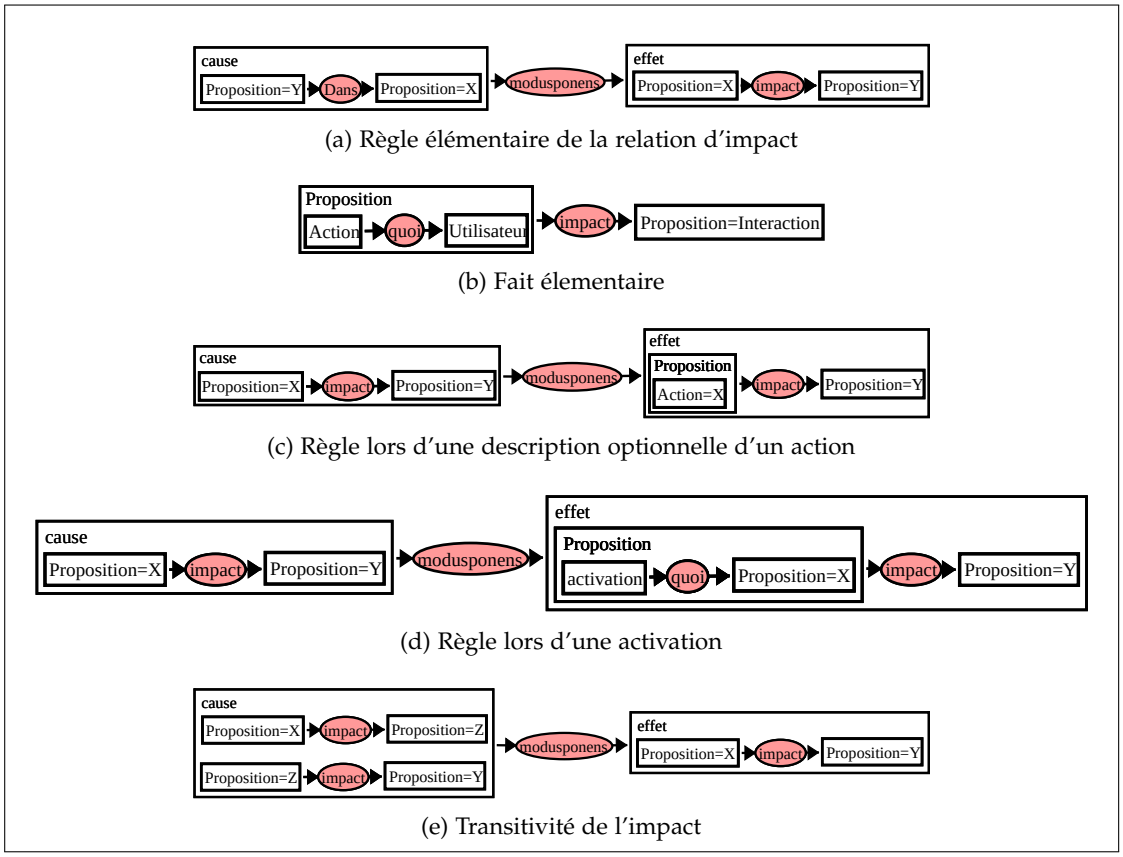
4.5.3.1 *Impact entre situations*

De manière similaire, l'*impact* peut déjà s'exprimer entre :
– un *outil* et un *nombre* : c'est l'*impact* initial de l'*usage* correspondant ;
– un *cas* et une *modification* : c'est l'ajustement de l'*impact* à la situation ;
– une *proposition* et un *nombre* : c'est l'impact global de la réaction décrite.

Mais il est appréciable de pouvoir comparer les situations entre-elles vis-à-vis de leur *impact* potentiel. Ainsi, les réactions modifiant l'interaction ont souvent un fort impact initial. Il peut être alors intéressant de diminuer cet impact lorsque la situation semble plutôt ordinaire, afin de permettre la réaction au moment où elle serait le mieux acceptée par l'utilisateur. Pour cela un *cas* permettant de définir cette *modification* est ajouté. Mais comment s'assurer que toutes les réactions que l'on considère comme affectant l'interaction soit considérée par l'*usage* du *cas* ? Il faut pouvoir exprimer qu'une situation a un impact sur une autre situation ou sur un concept.

Ainsi l'*impact* entre deux *propositions* est défini. Considérons que la première proposition contienne la description d'une réaction. Premièrement, cette réaction est considérée avoir un impact sur la situation et les sous-situations qu'elle décrit (à l'aide de la règle Fig. 77a). E.g le choix d'une technique d'interaction impacte les concepts d'interaction, de technique, de choix, de choix de technique etc. C'est en fait le comportement que l'on obtenait en utilisant directement un *cas*. Mais il est possible d'ajouter d'autres interprétations et de permettre au moteur de les combiner :
– l'explicitation directe d'un impact : agir sur l'utilisateur impacte l'interaction (Fig. 77b).
– la description d'une action est considérée pour évaluer son impact (Fig. 77c).
– le contenu d'une activation est considérée de même (Fig. 77d).
– la relation d'impact est transitive (Fig. 77e.

Cela permet par exemple de pouvoir déterminer que l'activation d'une modification de l'espace, décrite comme un déplacement de l'avatar a un impact sur l'interaction (comme pour une des assistances finales, Fig. 113a section 5.6).



(a) Règle élémentaire de la relation d'impact

(b) Fait élementaire

(c) Règle lors d'une description optionnelle d'un action

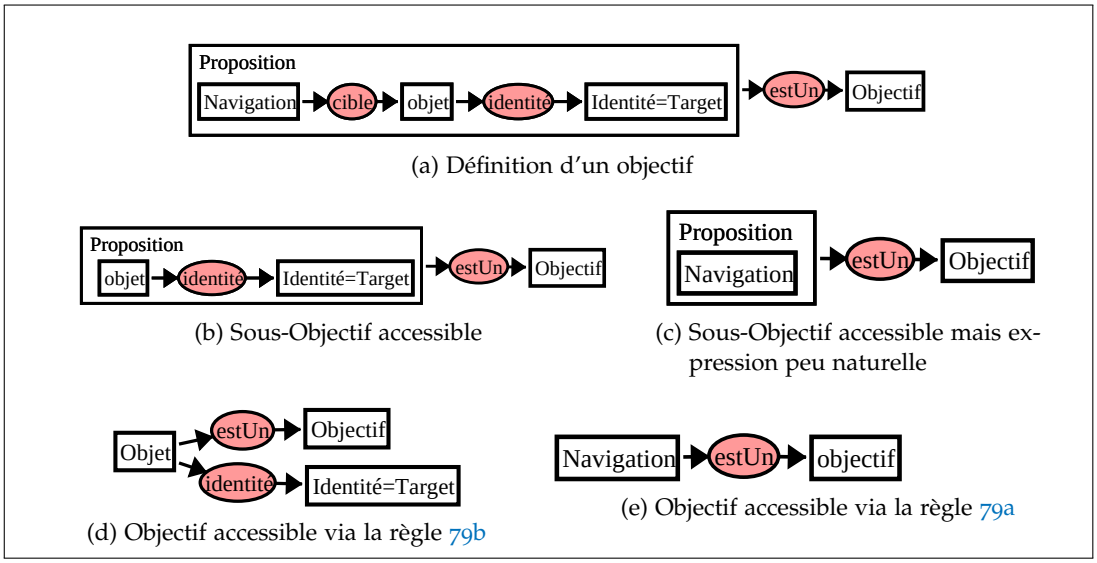
(d) Règle lors d'une activation

(e) Transitivité de l'impact

FIGURE 77: Exemples de définition des possibilités d'impact entre propositions

4.5.3.2 *Distributions d'information et expression naturelle*

L'expression naturelle se heurte parfois aux structures utilisées. Ainsi lorsque nous exprimons une situation où la navigation vers un objet donné est un objectif (Fig. 78a), nous aimerions pouvoir exprimer que les sous-situations compatibles sont elles aussi des objectifs. Ce qui est possible : en utilisant le même type de structure, Fig. 78b et Fig. 78c, on obtient qu'il est vrai que l'objet spécifique et la navigation sont des objectifs. Ces écritures ne sont en revanche pas forcément naturelles et on est tenté de vouloir les représenter directement telles que Fig. 78d et Fig. 78e. Pour permettre ces écritures à partir de la description globale, deux règles sont ainsi ajoutées (Fig. 79a et 79b)



(a) Définition d'un objectif

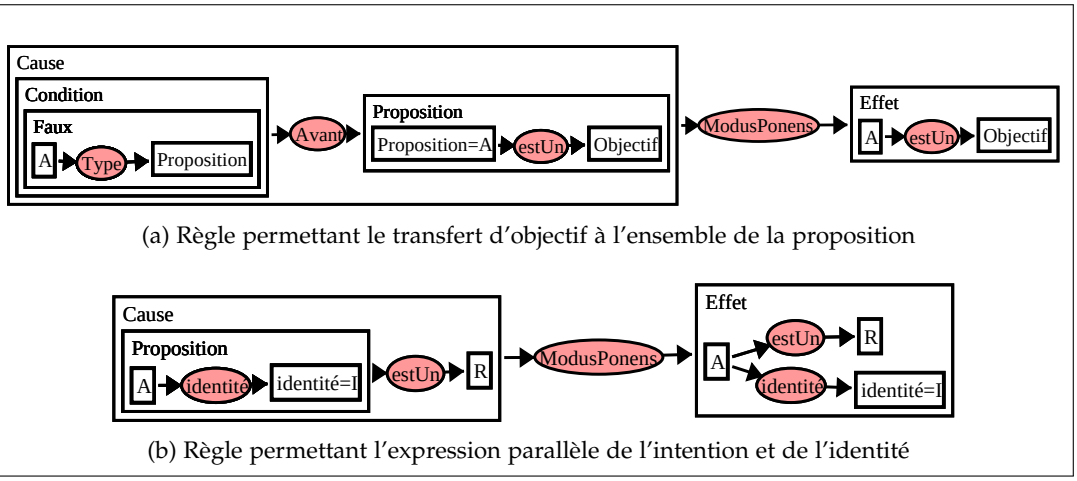
(b) Sous-Objectif accessible

(c) Sous-Objectif accessible mais expression peu naturelle

(d) Objectif accessible via la règle 79b

(e) Objectif accessible via la règle 79a

FIGURE 78: Exemples d'expressions d'objectifs



(a) Règle permettant le transfert d'objectif à l'ensemble de la proposition

(b) Règle permettant l'expression parallèle de l'intention et de l'identité

FIGURE 79: Exemples de règles pour l'expression naturelle d'un sous-objectif

Cette distribution de l'information à travers un concept englobant peut également être désirée pour répartir l'information sur plusieurs CGs, grâce au concept de *groupe*. Par exemple pour permettre à un capteur de définir des attributs pour un ensemble d'entités du même type simultanément. La description d'un *groupe* est ainsi toujours une liste. Il faut ensuite ajouter des règles pour gérer un éventuel transfert de propriétés du groupe à ses éléments. Par exemple pour gérer le transfert d'*attributs* d'un groupe à ses éléments Fig.80.

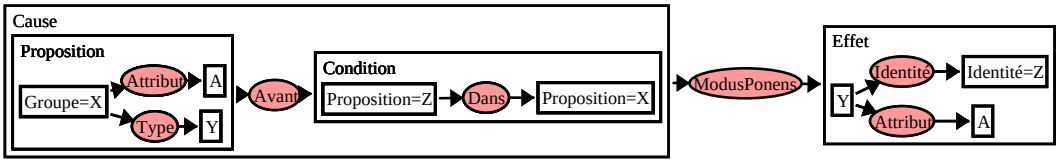


FIGURE 80: Transfert de propriété des groupes

De nombreuses règles similaires sont introduites pour permettre des expressions naturelles ainsi que des interprétations supplémentaires de CGs. Dans une version

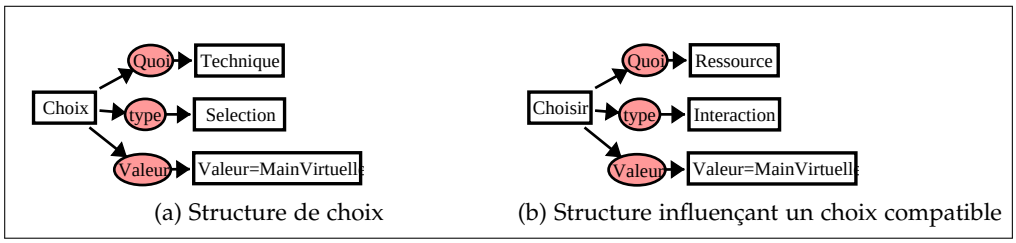
finale, il serait intéressant de ne pas les inclure à l'exécution mais de les utiliser uniquement pour formater les demandes et les faits ajoutés. En effet, la multiplication de ces règles est rapide.

4.5.4 *Formes de combinaison d'informations*

4.5.4.1 *Notion de choix*

Pouvoir effectuer un choix est intéressant. Le processus de réaction effectue automatiquement de nombreux choix notamment celui des réactions les plus adaptées. Cependant il est utile d'expliciter le concept afin d'éviter par exemple que les activations de deux techniques d'interaction différentes puissent être simultanément décidées.

Ainsi la notion de choix permet d'exprimer la possibilité à privilégier parmi plusieurs. Le CG Fig. 81a est une structure décrivant le choix d'une *valeur* spécifique pour une situation décrite via la relation *quoi*. Cette situation peut être également classée selon un *type*. Cette valeur est automatiquement choisie comme la meilleure parmi celles disponibles. La meilleure est obtenue en comparant les volontés existantes de *choisir* différentes *valeurs* (Fig. 81b). Ainsi, la confiance de chacun des CGs statuant sur l'action de *choisir* et possédant une description et un *type* compatible avec la demande initiale est comparée. Seule la valeur correspondant à la confiance maximale est attribuée à la notion de choix. Ainsi le choix combine la notion de confiance pour établir une notion de vérité.



(a) Structure de choix (b) Structure influençant un choix compatible

FIGURE 81: Structures permettent de gérer un choix

4.5.4.2 *Notion d'agrégation*

Il est intéressant de réfléchir séparément à chaque élément d'adaptation, comme la modalité ou l'action. Le moteur favorise ce raisonnement sémantique avec combinaisons d'idées et d'informations. Mais parfois des formes imposées empêchent d'exploiter ce raisonnement naturel. C'est-à-dire qu'il faudrait parfois sur-définir, en précisant également des concepts génériques non pertinents, uniquement afin de s'assurer de la compatibilité avec une structure de CG utilisée par la suite. Or on aimerait, tant que possible, être indépendant d'une de ces structures en particulier, le contenu éventuel des *usages* des *outils* de l'application. Cela permet de réduire les contraintes imposées pour l'usage du moteur. Il reste toujours possible pour le concepteur d'ajouter une règle pour rendre ses outils compatibles avec des situations déjà gérées ou d'adapter la forme de ces outils.

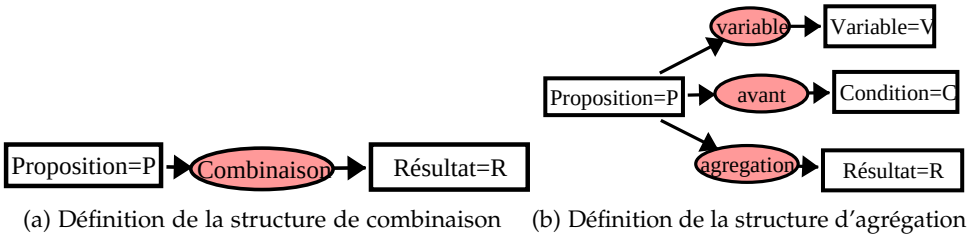
Mais c'est ainsi pour pouvoir générer des adaptations à partir d'éléments de réflexion que l'*agrégation* a été ajoutée. Elle permet de générer la structure correspondant aux usages des outils sans perdre en sémantique. L'agrégation est ainsi

utilisée pour former des descriptions complètes de réactions potentielles à partir d'étapes naturelles liant des possibilités de réaction élémentaires (section 5.5 au chapitre suivant).

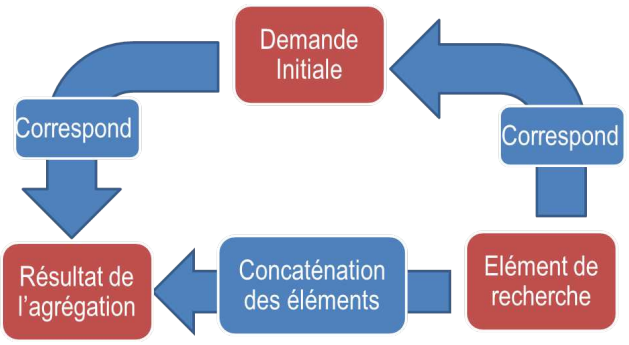
L'agrégation se base en interne sur la jonction maximale de CGs, que l'on peut obtenir également explicitement via le concept de *combinaison* qui concatène une liste de CGs (Fig. 82a). La liste de la *proposition* est combinée en concaténant deux-à-deux les CGs suivant leurs branches communes et en tentant de garder le plus d'éléments spécifiques à chacun. La combinaison est stockée dans le *résultat*. Cependant, il n'est pas possible de concaténer de manière libre des CGs. En effet, les graphes ne sont pas tous compatibles entre-eux et il faut donc avoir au moins un concept commun. De plus, certaines combinaisons seraient des fautes de raisonnements (e.g. obtenir un *objet* avec deux *identités* différentes).

Mais souvent la liste de CGs à agréger est dépendante de la base de connaissance. Ainsi une structure définit le processus d'agrégation (Fig. 82b). Dans la *proposition* est un CG qui représente un motif de recherche. La partie du motif qui correspond aux solutions qui seront agrégées est indiquée par la *variable*. Des *conditions* éventuelles peuvent être précisées et alors vérifiées pour chaque solution. L'ensemble des solutions, vérifiant également les conditions, sont alors agrégées et le *résultat* stocké. La confiance associée à cette structure doit être interprétée comme celle du résultat. Cette confiance est égale à la moyenne des confiances des éléments d'information utilisés pour l'agrégation.

Un autre avantage de cette structure est de s'assurer souvent implicitement que les CGs soient combinables puisque issus d'un même motif. Mais des tests supplémentaires s'imposent la plupart du temps. En pratique, les règles l'utilisant le font via un principe similaire (Fig. 82c). Il faut que les éléments recherchés correspondent, d'une manière définie dans la règle, à la demande initiale. Ainsi on ne récupère que des éléments intéressants. La récupération se fait grâce à la définition du motif de recherche et des conditions. Le résultat obtenu n'est alors pas toujours suffisant pour répondre à la requête. La règle ajoute alors en général un test sur le résultat de l'agrégation.



(a) Définition de la structure de combinaison (b) Définition de la structure d'agrégation



(c) Principe courant des règles appelant ces structures

FIGURE 82: Illustration de la notion d'agrégation

4.5.5 *Conclusion sur l'extension des concepts*

Ainsi ces nombreux concepts et règles peuvent être réutilisés par la suite dans nos applications. En pratique, c'est l'inverse qui s'est produit. Chaque partie d'application a, au départ, nécessité une création spécifique concernant la base de connaissance. Les concepts les plus généraux ont été ainsi réunis dans cette section. Mais cela illustre globalement le raisonnement sémantique et sa réutilisation. La base de connaissance se forme peu à peu et chaque morceau peut être séparé. Selon l'usage certaines notions peuvent ou non être chargées. Mais idéalement l'ensemble se combine peu à peu et permet la gestion d'un raisonnement complexe global issu de ces pièces élémentaires.

4.6 CONCLUSION SUR L'APPORT DU MOTEUR

Le moteur permet un raisonnement sémantique qui peut être directement modifié, étendu et adressé dynamiquement par un utilisateur, un concepteur ou une application. Implémenté avec la plateforme Amine, on peut y accéder depuis Prolog+ CG. Il peut être également utilisé comme un thread Java et alors communiquer avec le protocole OSC pour recevoir ou transmettre des informations. Le moteur utilise les graphes conceptuels, basés sur une ontologie, qui permettent la description de toutes situations en reliant des concepts entre eux. Ces graphes possèdent une sémantique à la fois lisible par l'utilisateur et la machine. A l'aide des opérations gérant ces graphes, ainsi que de logique du premier ordre, des processus sont créés permettant de gérer de nombreuses notions.

Ainsi, un méta-interpréteur est créé afin d'étendre la notion de vérité, d'en intégrer une notion de degré (la confiance) ainsi que la gestion du temps. La notion de vérité permet de trouver les solutions à une requête tout en bénéficiant automatiquement dans notre raisonnement sémantique de la péremption des évènements et

de la mise-à-jour d'un historique. La confiance, elle, permet initialement d'exprimer des nuances pour chaque situation et chaque étape de raisonnement. De plus, la confiance peut également être calculée quelle que soit la sémantique de la requête et permet alors d'obtenir une quantification englobant l'ensemble des informations et raisonnements compatibles avec la demande. Cela permet de représenter l'adéquation d'une situation décrite à la base de connaissance. Mais cela permet également de comparer des situations à la sémantique différente.

Ainsi, à l'aide de ces notions, un processus de réactions automatiques peut être demandé. Il permet de gérer l'ensemble des réactions potentielles grâce à la connaissance des outils qui peuvent être commandés par le moteur. Ces outils sont décrits et ajoutés par l'application. Ils ont notamment un usage, une commande et un impact. La sémantique de l'usage permet de calculer la confiance de son déclenchement au vu de la situation. Le concept d'impact quantifie le degré de répercussions sur l'utilisateur et peut être également modifié par le contexte. Les notions d'impact et de confiance permettent au moteur de classer les réactions disponibles et leur pertinence. Il peut alors sélectionner les meilleures tout en limitant leur impact total. Un impact total admissible permet ainsi d'éviter de saturer l'utilisateur de réactions. L'ensemble permet d'obtenir un raisonnement sémantique automatique conscient du contexte.

L'avantage principal du moteur est sa grande expressivité et ses possibilités d'accès et de modifications tant pour les informations gérées que pour son raisonnement. Il est également sémantique et alors plus facile à comprendre, à réutiliser, à modifier, à créer et à interroger. Certaines propriétés des processus fondamentaux sont également paramétrables et modifiables de façon similaire. Des règles sont également créées dans le but principal d'étendre les capacités du moteur. Finalement, il est possible d'utiliser le moteur pour de nombreux buts, à la fois la compréhension de situations et/ou le déclenchements de réactions automatiques. Désormais le moteur va être utilisé pour obtenir nos assistances automatiques à l'interaction 3D. Des bases de connaissance ainsi que d'outils pour nos applications de réalité virtuelle sont alors créées, en s'appuyant sur l'ensemble des capacités et concepts déjà décrits.

5

CRÉATIONS D'OUTILS ET D'UNE BASE DE CONNAISSANCE POUR L'ASSISTANCE À L'INTERACTION 3D

Résumé : *Ce chapitre présente la création des outils et des règles permettant d'obtenir des assistances adaptatives pour nos applications. C'est à la fois un exemple, un modèle du processus entrepris par le concepteur et la présentation des bases du raisonnement qui mènent aux applications du chapitre suivant. Premièrement est présentée la création de capteurs et de règles permettant d'obtenir des informations initiales. Parmi ces informations, l'obtention d'indices menant aux intérêts et objectifs de l'utilisateur est primordiale. Ensuite, des étapes sémantiques supplémentaires sont introduites, lorsque possible, décrivant des situations obtenues par des règles de combinaison. Celles-ci isolent les informations primaires de leur usage, facilitent leur ré-emploi et organisent le raisonnement. Cette abstraction favorise la fusion d'information qui peut être un but en soi. Finalement, des effecteurs sont créés puis classifiés pour obtenir des descriptions sémantiques adaptées au déclenchement d'assistance. L'ensemble des informations et situations est utilisé pour obtenir l'assistance automatique désirée.*

5.1 LIGNES DIRECTRICES POUR LA CRÉATION D'UNE BASE DE CONNAISSANCE ET SON EXPLOITATION

Sur les bases du moteur, il convient désormais d'utiliser les concepts et possibilités afin de décrire les réactions, dont les assistances via des adaptations, à un contexte. La création d'une base de connaissance, contenant notamment les règles, est un processus somme toute complexe. Il convient de rester suffisamment général, tout en étant efficace, afin de prévoir les différents usages et écritures des informations qui pourront être apportées par la suite (section 5.1.1). Pour obtenir des informations de l'application et pour pouvoir réifier des décisions du moteur, il faudra créer un ensemble d'outils pour notre gestionnaire d'environnement virtuel. Cette boite à outil pour notre gestionnaire doit être également réfléchie (section 5.1.2). D'ailleurs, les deux processus de créations ne sont pas complètement indépendants (section 5.1.3). L'utilisation devient ainsi en partie spécifique à l'application considérée. Cette facilité de personnalisation du raisonnement sur les bases du moteur est un des buts recherchés. De plus, des parts de raisonnement et la plupart des outils créés sont facilement réutilisables. C'est le cas des notions présentées précédemment (section 4.5), mais également de la plupart des situations (section 5.4) et assistances (section 5.6).

5.1.1 *Création des règles*

5.1.1.1 *Définition de différentes règles*

De nombreuses règles ayant des buts différents peuvent être créées. Certaines influencent le noyau du moteur, d'autres définissent des notions réutilisables et des extensions du raisonnement originel. Dans ce chapitre les règles visant la compréhension et l'assistance à l'interaction 3D sont considérées. Elles doivent profiter des possibilités du moteur afin de ne pas se contenter de liens directs entre capteurs et effecteurs, qui freinent la réutilisation et qu'il serait alors probablement préférable d'intégrer au sein de l'application. Ainsi, une première série de règles tente de décrire au mieux des étapes de l'adaptation à l'interaction 3D. Celles-ci permettront alors la fusion, via le moteur, de l'ensemble des informations pouvant correspondre à la situation et d'obtenir une meilleure assistance. Les ajouts futurs, notamment issues de capteurs supplémentaires, peuvent être exprimés comme contribuant à ces étapes, évitant de devoir recréer toute la chaine de raisonnement. Ensuite, des règles permettent le déclenchement de réactions. Celles-ci doivent également être pensées pour être tant que possible indépendantes des outils. C'est l'étape qui apporte le plus de subjectivité de la part de l'application. Mais partant des assistances sémantiques maitrisées, l'ajout d'une règle supplémentaire permet aisément de faire le lien avec des outils très particuliers ou exprimés autrement.

Ainsi, pour chaque application, le concepteur peut ajouter ses propres règles ou n'introduire que la partie nouvelle du raisonnement, en s'appuyant sur des réalisations précédentes. L'utilisateur peut lui-même ajouter des règles et faits pour décrire ses préférences, voire un comportement propre (même si cela nécessite vraisemblablement une interface spécifique non présente actuellement). Cette dernière utilisation peut être introduite par le concepteur via des outils pour présenter des possibilités issues du moteur à l'intérieur de l'environnement virtuel. L'utilisateur peut alors également faire des choix sémantiques guidés par les outils définis par le concepteur.

5.1.1.2 *Principe et difficultés de l'écriture des règles*

Les règles sont, tout comme l'ontologie, en mouvement perpétuel. Chaque règle doit être le plus générale possible, ce qui permet de limiter leur nombre et de décrire des situations claires. En revanche il faut dissocier tant que possible les différentes étapes éventuelles du raisonnement, pour permettre de bonnes combinaisons et une modification aisée à chaque étape. Il faut idéalement réussir à extraire la quintessence du raisonnement à l'aide d'étapes élémentaires. Cela explique la mouvance des descriptions de structures et de règles. La combinaison des approches descendantes, issues des notions existantes, et de l'approche ascendante pour la gestion de spécificités de l'application impose des réformes récurrentes. Les étapes introduites permettent alors de limiter tant la fréquence que la portée de ces modifications. Le nombre de changements nécessaires se réduit donc au fur et à mesure que le raisonnement est mis en place. Ainsi, en pratique, l'expression de règles générales nécessite de séparer ce qui peut l'être et multiplie ainsi les étapes intermédiaires.

5.1.2 *Création d'une boite à outils*

5.1.2.1 *Design des outils*

La création d'une boite à outils pour une application suit les mêmes principes que la création du moteur. Elle doit être tant que possible réutilisable, extensible et facile à connecter (bien que limitée à un gestionnaire d'environnement virtuel particulier dans notre cas). Chaque outil peut être implémenté de manière générale, afin de répondre à plusieurs besoins. Ceux-ci peuvent être différenciés au sein du moteur par l'inscription de plusieurs *usages* sémantiques liés à leur *identité*. Quelle qu'en soit l'implémentation, ils suivent un design particulier pour être commandés par le moteur, ou plus généralement pas des messages extérieurs. Parmi les outils présents dans l'application peuvent se trouver des effecteurs contrôlés par le moteur (Fig. 83c) et des capteurs contrôlés (Fig. 83b) ou non (Fig. 83a). Des effecteurs peuvent être utilisés comme capteurs semi-contrôlés (Fig. 83d). Un effecteur non contrôlé n'aurait aucun lien avec le moteur et n'est donc pas représenté. L'usage de ces outils se déroule ainsi :

- La description des outils commandés est ajoutée via un CG dans la base de connaissance du moteur à l'initialisation du module. Cette initialisation peut être faite à tout moment comme pour les autres ajouts sémantiques.
- Le processus de fonctionnement des outils commandés active un récepteur. Celui-ci permet d'obtenir par la suite les messages destinés à l'outil. Il est contacté grâce à son identité et chaque message contient une commande unifiée par le moteur.
- Les capteurs et certains effecteurs envoient une information à l'issue de leur processus. L'information doit être un CG sauf dans le cas des capteurs contrôlés. Les capteurs non contrôlés n'effectuent que cette étape.

Un module au sein de l'application peut inscrire à tout moment un CG dans la base de connaissance, indépendamment du moteur. Ce type de module représente un capteur non contrôlé (Fig. 83a). Au contraire un capteur contrôlé nécessite une commande de la part du moteur (Fig. 83b). Ce module est le seul décrit comme un *capteur* dans la base de connaissance et correspond à l'usage de *questions*. Il retourne une information via un canal spécifique aux *capteurs*. Celle-ci correspond à sa commande initiale complétée par ses informations propres. Un avantage de ce fonctionnement est que cette commande complétée n'a pas à être sémantique. Le moteur est capable de l'interpréter grâce à la description de l'outil. De plus, le contexte de la question originelle a été sauvegardé par le processus de décision. L'information ajoutée à la base est ainsi dépendante du raisonnement qui a amené la question.

Un *effecteur* a pour premier usage d'affecter l'application de manière perceptible pour l'utilisateur. Mais le moteur n'a pas connaissance de sa finalité. Ainsi un *effecteur* correspond pour le moteur à la gestion de l'envoi d'une *commande* externe selon une situation d'*usage*. L'application réceptionnant la commande du moteur déclenche en fonction d'elle un processus qui a donc souvent un effet perceptible (Fig. 83c). Les effets peuvent être multiples. À l'issue de ce processus, un effet optionnel habituel est une mise à jour de la base de connaissance : cette partie de l'effecteur agit alors comme un capteur semi-contrôlé. L'ajout d'une information

suite à un déclenchement n'est pas attendue par le moteur.

Il est alors possible de créer des *effecteurs* [1] dont le but au sein de l'application est d'obtenir cet envoi d'information sémantique. Ces modules sont appelés des capteurs semi-contrôlés (Fig. 83d). C'est un choix d'implémentation qui n'utilise pas le canal de retour de *questions*. C'est intéressant lorsque le contexte de la question n'est pas important ou lorsque la réponse n'est pas toujours possible. Mais son usage habituel est l'activation d'une source d'information. Le processus au sein de l'application reste ainsi actif et met à jour automatiquement la base de connaissance sans contrôle supplémentaire du moteur. Le moteur connait néanmoins son état et peut gérer sa désactivation. Cela permet d'éviter au moteur de devoir constamment poser la même question à chaque cycle de réaction, lorsque l'information reste intéressante à obtenir.

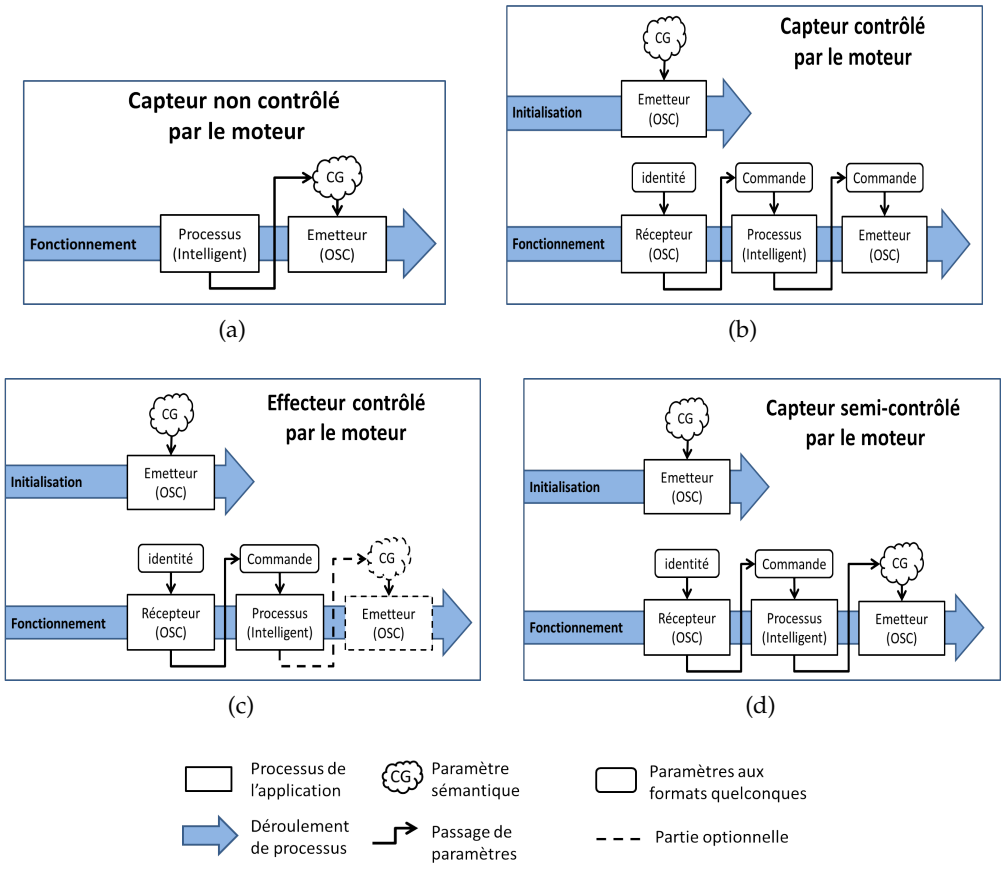


FIGURE 83: Designs des outils

Les récepteurs peuvent être directement capables de gérer le protocole OSC : le moteur est alors utilisé pour émettre chaque commande sur un canal spécifique via l'identité de l'outil inscrite dans sa description. Autrement le récepteur peut être spécifique à l'application et dans ce cas la gestion des commandes issues du moteur est centralisée par l'application. Un message contenant l'ensemble des commandes des outils doit être alors récupéré grâce au protocole OSC et décomposé pour

1. Cela peut être des *outils* sans plus de précision mais ils ne bénéficient alors pas de la sécurité d'unification complète de leur commande.

transmettre chaque commande selon le protocole interne à l'application.

L'inscription des informations peut se faire via l'usage de fonctions PCG prédéfinies effectuant une part du formatage automatiquement et vérifiant les données inscrites. L'appel à une fonction spécifique est obligatoire pour le retour d'un capteur contrôlé qui n'est pas forcément un CG. Autrement des bloc émetteurs typiques peuvent être créés par convenance, potentiellement chacun sur un canal OSC séparé et appelant une fonction spécifique. Des blocs gèrent ainsi selon leurs entrées l'ajout de faits spécifiant ou non leur date et leur confiance. L'appel à ces fonctions permet par exemple de numéroter et de dater automatiquement les faits introduits. D'autres blocs permettent de commander au moteur la suppression complète ou la péremption (et donc le passage dans l'historique) d'une description. Néanmoins l'ensemble des communications pourrait être géré avec uniquement le bloc émetteur basique permettant l'évaluation d'une expression PCG par le méta-interpréteur (en précisant manuellement la *supposition* des faits par exemple). En revanche, utiliser des blocs spécifiques au sein de l'application permet de gérer des messages internes spécifiques en plus des effets automatiques de formatage. Cela permet par exemple de déclencher une réaction automatique selon les ajouts d'information (e.g après chaque évènement) ou après chacun d'entre eux.

5.1.2.2 *Conséquence du design de l'outil*

L'outil tel qu'il est défini est un processus capable d'être déclenché par un message. La plupart génèrent également une information sémantique à partir des résultats du processus (souvent en remplissant une structure CG préformée). Finalement, le design de l'outil impose plusieurs propriétés à l'application les utilisant qui doit :

- gérer la modularité de ces réactions. Des raisonnements doivent être isolés pour correspondre aux outils et pouvoir être déclenchés en parallèle. C'est peu contraignant pour les approches composants.

- créer leur définition sémantique. Il faut décrire principalement le ou les usages qui doivent résulter en une commande de l'outil. Néanmoins la commande n'est pas forcément sémantique et peut être notamment élaborée en fonction de variables pertinentes de l'*usage* (e.g. ne garder que les termes pertinents pour le processus interne comme l'*identité* d'une cible).

- gérer la commande par message. Les modules correspondants aux outils doivent pouvoir être commandés par des messages. Le protocole OSC doit aussi être implémenté de manière centralisée ou non dans l'application. Il faut potentiellement prévoir l'effet d'un message perdu ou répété.

- choisir le type de capteur pour chaque information. Un capteur indépendant, seul ou en fin d'effecteur contrôlé, peut être plus intéressant. Un capteur semi-contrôlé limite les allers-retours entre l'application et le moteur, ce qui est appréciable pour la rapidité du raisonnement et la fraicheur du contexte. La description via un capteur contrôlé se focalise sur la question posée et l'information à obtenir. Cette forme permet également de continuer partiellement le

raisonnement qui a amené la question.

– choisir le type d'informations envoyées et gérées pour chaque capteur. Les informations peuvent être des faits ou des évènements (voire des règles). Un fait est utile si sa validité est permanente ou durable et gérée par la suite. Un événement est utile lorsque le temps est important ou lorsque le capteur n'est pas capable de gérer la péremption de ces informations. Fournir une confiance ou non est également un choix. Il dépend principalement de la capacité à fournir des degrés dans l'information. Enfin, les outils peuvent gérer de manière plus ou moins avancée les informations du moteur notamment les leurs. Ainsi un capteur peut avoir la maitrise complète de ses informations en ajoutant la relation *source* à ses *faits* menant à son *identité*. Il peut alors par exemple déclencher lui-même la suppression, la péremption ou tout autre demande en bénéficiant des capacité d'unification du moteur.

5.1.3 *Synergie de créations*

La création de comportements n'est pas facile à isoler. L'approche est une succession d'analyses et de synthèses. La conceptualisation initiale est influencée par les besoins d'expressions. L'usage concret de cette conceptualisation effectuée modèle alors les règles et outils. L'apparition d'usages non ou difficilement gérés modifie en retour la conceptualisation. En pratique ontologie, structures de CGs, raisonnements implémentés, outils et descriptions d'outils évoluent ensemble. C'est pourquoi une représentation et un raisonnement expressifs, sémantiques et facilement modifiables sont intéressants. Et c'est aussi une des raisons premières de la difficulté de la création de ce moteur, indépendant des applications. Les étapes de raisonnements introduites permettent des évolutions graduelles et encadrées.

5.2 OBTENTION D'INFORMATIONS POUR L'IDENTIFICATION DE LA SITUATION

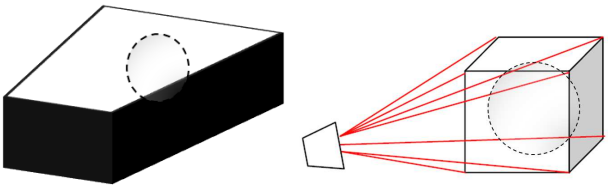
La première étape pour obtenir une assistance intelligente est d'avoir des informations à partir desquelles réfléchir. Or les applications qui communiqueront avec le moteur, a priori, ne décrivent pas un environnent sémantique où ces informations seraient plus aisées à obtenir. L'application doit offrir un moyen d'extraire ces informations, à l'aide de capteurs, contrôlés ou non par le moteur. Tout d'abord, tenter d'estimer l'intention de l'utilisateur est primordial et a mené à la construction de nombreux capteurs, détaillés dans leur propre section 5.3. Des attributs concernant des objets de l'environnement peuvent être mesurés (section 5.2.1). D'autres informations diverses peuvent être utiles comme connaitre l'initiateur de la demande de réaction, l'utilisateur ou l'application. (section 5.2.3).

5.2.1 *Obtention d'attributs*

Les *attributs* des objets de l'environnement virtuels peuvent contribuer à la discrimination de la situation. De multiples *attributs* sont donc considérés et leur obtention réunie dans un même capteur.

5.2.1.1 *La visibilité*

Un objet peut être *visible* ou *caché*. La détermination de cet *attribut* utilise premièrement l'information de présence ou non de l'objet dans le frustrum de la caméra (Fig. 84a). Cette information de présence dans la zone affichable ne garantie pas seule la visibilité : un objet peut être caché par un obstacle. La deuxième information utilisée est le ratio du nombre de rayons virtuels n'intersectant aucun obstacle, entre différents sites d'émissions sur la boite englobante de l'objet et le centre de la caméra (Fig. 84b). Elle correspond à la confiance que l'objet soit *visible* dans le cas où il est affichable. Si l'objet n'est pas affichable, cette confiance est automatiquement nulle. En effet si l'objet est derrière la caméra, la visibilité donnée par le ratio seul pourrait être faussement élevée. Ce ratio peut être considéré en soi comme le potentiel de visibilité vis à vis d'une rotation éventuelle de la caméra. Il représente alors la notion de *faible couverture* par des obstacles de l'objet. Un objet à faible couverture et non visible peut ainsi être rendu visible par une modification de la direction de la caméra.



(a) Visibilité binaire (b) Degré de visibilité potentiel

FIGURE 84: Principe de gestion de la visibilité d'un objet

5.2.1.2 *L'entourage*

L'*entourage* est une mesure de la quantité d'entités entourant un objet. Sur le même principe que précédemment, une série de rayons sont émis de la boite englobante de l'objet. Pour une longueur donnée, le ratio de rayons intersectant des obstacles donne une mesure de l'entourage de l'objet. Ce ratio correspond directement à la confiance en l'expression d'un attribut d'*entourage élevé*. Il est possible en choisissant les groupes de collision de distinguer les obstacles bruts des autres objets manipulables.

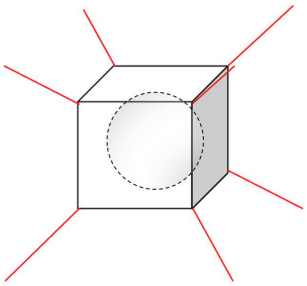


FIGURE 85: Principe de gestion de l'entourage d'un objet

5.2.1.3 *L'éloignement*

La *distance* entre l'objet et la caméra active est mesurable. A cet éloignement, contrairement aux deux précédents attributs, correspond une mesure continue. La confiance en l'expression d'un attribut de *distance élevé* est donc obtenue via la

transformation de la valeur de la distance par une fonction (Fig. 86). Cette fonction est non linéaire afin de refléter des impressions subjectives. Un objet nous apparait ainsi qualitativement également proche ou éloigné dans les situations extrêmes. De même un palier central permet de gérer qualitativement la notion d'un objet à distance moyenne. Ces paliers correspondent à des paramètres d'entrées de l'implémentation afin de pouvoir gérer des notions de distances différentes en fonction des situations d'applications.

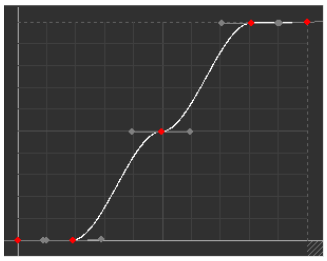


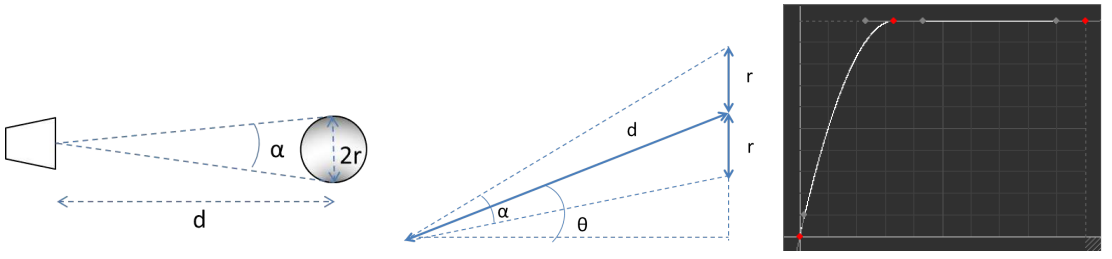
FIGURE 86: Gestion de confiance non linéaire pour la notion continue de distance élevé

5.2.1.4 *La visibilité angulaire*

Une description spécifique de la *visibilité* est l'*angle* sous lequel est vu l'objet. Cela permet d'avoir une mesure du ratio de l'objet à l'écran. Cette mesure est obtenue qualitativement par $\frac{r}{d}$, le rapport du rayon de la sphère englobante à l'objet par rapport à la distance à la caméra (Fig. 87a). C'est une approximation au premier ordre (avec $\theta \ll 1$ et $d \gg r$) du cas général (Fig. 87b) :

$$\alpha = \arctan(\frac{d\sin(\theta) + 2r}{d\cos(\theta)}) - \arctan(\frac{d\sin(\theta) + r}{d\cos(\theta)})$$

L'approximation est amplement suffisante pour un ordre de grandeur. En effet on ne considère déjà que le rayon apparent de la sphère englobante, de plus la confiance en l'attribut d'un *angle* de *visibilité élevé* est obtenue qualitativement par une fonction non linéaire (Fig. 87c). Cette fonction permet de modifier plus fortement la confiance dans une première phase ou l'objet est à peine visible. Par la suite, continuer à augmenter la séparation angulaire n'impacte plus autant la différence ressentie qualitativement quant à la visibilité de l'objet.



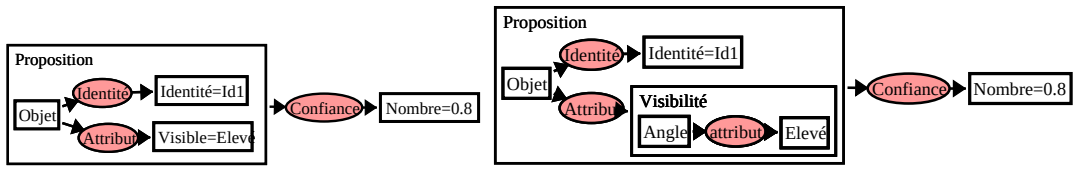
(a) Illustration : taille et distance (b) Calcul du ratio (c) Confiance non linéaire en l'attribut : ratio élevé

FIGURE 87: Principe de gestion de la séparation angulaire et de la distance d'un objet

5.2.2 *Discussion sur l'usage des attributs*

5.2.2.1 *Influence du choix des descriptions d'attributs*

La structure récurrente d'expression de ces mesures (e.g. Fig. 88a) a été dans nos applications la relation d'*attribut* entre un *objet* d'*identité* connu et un *qualificatif* (e.g. *visible*, *entourage*, *distance*, *couverture*, *visibilitéAngulaire*[2] etc.) décrit par un *quantificatif* (e.g. *faible* ou *élevé*). Il est en effet intéressant de pouvoir séparer chaque influence.



(a) Type de CG retourné par le capteur d'attribut

(b) Évolution considérée des informations d'attributs

FIGURE 88: Choix de description des attributs

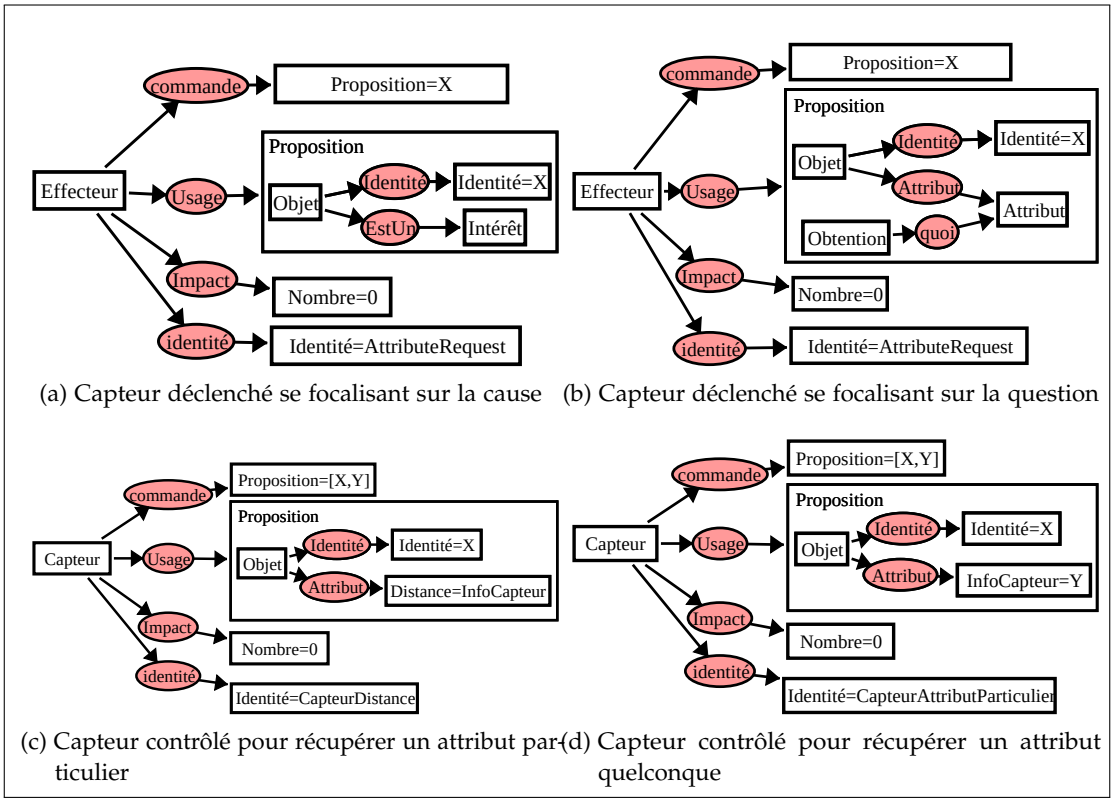
Mais il sera préférable de faire évoluer cette représentation notamment pour qualifier la *visibilité* (e.g. Fig. 88b). Celle-ci peut avoir plusieurs facteurs, décrits comme un CG emboité l'expression de :

- une *visibilité directe* plutôt *élevée* lorsque l'objet est dans le champs de vision et peu couvert (anciennement objet *visible*).
- une *visibilité* avec un *potentiel élevé* (anciennement la *couverture faible*)
- une *visibilité* avec un *angle élevé* (anciennement *visibilitéAngulaire*)
- d'autres facteurs de visibilité comme la couleur, le mouvement, etc.

L'avantage est de pouvoir alors obtenir la confiance en une visibilité élevée qui combinent ces différents facteurs directement (sans ajouter de règle) en profitant mieux de l'outil CG. Déduire qu'un objet est visible lorsque la visibilité de type directe est présente permet de garder l'expression naturelle précédente. La confiance en un objet visible peut alors être lié à la visibilité sans en altérer le sens (un objet n'est visible que si cela est exprimé directement ou dans le cas d'une visibilité directe). Il faut pour cela ajouter une unique règle déduisant un a priori sur la confiance d'un objet visible lors d'une situation de visibilité élevée. L'expression est naturelle et les détails de la visibilité restent accessibles au besoin.

2. L'apparition d'un terme composite est souvent le signe d'une nécessité de refonte.

5.2.2.2 *Définition des descriptions sémantiques de l'outil*



(a) Capteur déclenché se focalisant sur la cause

(b) Capteur déclenché se focalisant sur la question

(c) Capteur contrôlé pour récupérer un attribut particulier

(d) Capteur contrôlé pour récupérer un attribut quelconque

FIGURE 89: L'outil gérant l'ensemble des requêtes d'attributs

De nombreuses descriptions de ce capteur d'attributs sont possibles. Par exemple, l'usage d'un effecteur contrôlé, qui se focalise sur la situation le déclenchant (Fig. 89a) ou sur le type d'information à obtenir (Fig. 89b). Il est également possible de le faire via un capteur pouvant être appelé pour tout attribut (Fig. 89d) ou un attribut en particulier (Fig. 89c). Il est d'ailleurs aisé d'implémenter plusieurs comportements dans le même bloc de l'application, avec plusieurs définitions sémantiques (notamment pour les dernières descriptions, une par attribut géré).

L'implémentation effective est la première (Fig. 89a). La description de type capteur contrôlé n'est pas utilisée car l'attribut est directement inscrit dans la base et ne subit pas de traitement particulier en fonction de la partie de raisonnement l'appelant. D'ailleurs l'implémentation comme un capteur d'attribut général, doit prévoir un cas par défaut, pour une demande d'attribut qui ne serait en fait pas géré par ce capteur. L'ensemble des attributs est donc commandé via un même outil. Cette implémentation est déclenchée dès lors qu'un objet a été détecté comme un intérêt. Ce capteur récupère alors un ensemble d'informations (ici les attributs). Chaque attribut est ajouté de manière séparée car cet *outil* regroupe en pratique un ensemble de capteurs indépendants. Cela nous permet dans les premières applications de mettre à jour facilement le capteur, sans modifier la description, quelles que soient les informations que l'on aimerait obtenir pour qualifier un intérêt.

A cela peut également s'ajouter la notion d'activation. Activer le capteur permettrait de continuer à suivre les objets d'intérêts (tant qu'ils restent plutôt intéressants). Mais cela nécessite une implémentation quelque peu différente afin de mémoriser

les objets suivis. Une première option d'activation est de n'envoyer qu'une seule fois les attributs afin éviter une commande du capteur tous les cycles de réactions. Cela peut être gênant pour la fraicheur des informations car une mobilité quelconque les fait évoluer. La deuxième option est que le capteur activé soit capable de détecter un changement d'attribut et de ne mettre l'information à jour que dans ce cas. C'est une forme de capteur très intéressante à mettre en place et qui est utilisée dans le cas de l'étude de mouvement notamment. Enfin sans activation, tant que l'objet est un intérêt, la demande d'attribut est actuellement effectuée à chaque cycle de réaction.

5.2.3 *Autres sources de sémantique*

5.2.3.1 *Informations sémantiques stockées dans un objet*

Un objet peut stocker au sein même de l'application des informations sémantiques. Par exemple, à propos de son usage, de son historique d'usages ou de modifications, etc. Cela permet de gérer dynamiquement des informations spécifiques plutôt que de les charger toutes simultanément dans le moteur à l'initialisation de l'application. C'est une approche plus facile à soutenir pour des environnements de très grande taille. Le moteur peut ainsi récupérer au besoin, comme pour les attributs précédents, les informations sémantiques disponibles. Cet outil n'a pas été réellement utilisé pour deux raisons. Premièrement, l'effort porte principalement sur l'extraction de ces informations dans le cas d'un environnement non sémantique, sans connaissance préalable de telles informations. Enfin, les tests ont porté sur des environnements de faible dimension : si l'on veut tester l'ajout d'une information sémantique liée à un objet, l'ajout peut se faire directement dans la base de connaissance.

5.2.3.2 *Informations sémantiques issues d'effecteurs mixtes*

De nombreux effecteurs émettent des informations sémantiques en supplément de leur fonction principale. Par exemple, la reconnaissance de gestes peut en plus du geste détecté ajouter automatiquement sa direction. C'est particulièrement utile lorsqu'un effecteur activé possède plusieurs états de fonctionnement différents. Par exemple, l'effecteur d'attraction, après activation visant un objet support, n'applique une force sur la cible que si elle est suffisamment proche de l'objet support. L'effecteur peut alors indiquer la présence ou non d'une force actuellement appliquée par cet objet. En effet, cette information est alors conditionnée mais pas directement déductible de l'activation précédente de l'attraction.

5.2.3.3 *Informations sémantiques de fonctionnement*

Des informations sur le fonctionnement même du moteur sont utiles, notamment pour pouvoir le paramétrer automatiquement. Par exemple, savoir que l'utilisateur est perturbé par les réactions appliquées serait pertinent. Cela est difficile à obtenir en toute généralité sans retour direct de l'utilisateur mais est envisageable pour certaines réactions (e.g. attraction). Néanmoins parmi les informations générales pertinentes touchant le moteur que l'on peut obtenir se trouvent :

- le type de demande de réaction : automatique issue de l'application directement ou manuel avec l'utilisateur pour origine.

– le temps d'une réaction ou entre deux réactions : possiblement obtenues directement depuis le moteur.

5.3 INDICES POUR DÉTECTER L'INTENTION DE L'UTILISATEUR : INTÉRÊTS ET OBJECTIFS

Dans le cas d'un environnement collaboratif, sans connaissance a priori sur les participants, étudier le comportement de ces derniers permet d'élaborer des hypothèses sur les activités et sur le rôle de chacun dans la tâche. Cependant en se concentrant déjà sur un seul utilisateur, sans hypothèse initiale, l'étude de son comportement permet d'obtenir des indices sur son activité et son intention. Le principal rôle à extraire est alors celui des entités de l'environnement, vis-à-vis de l'activité de l'utilisateur. Savoir qu'un objet est un *intérêt*, ou même un *objectif*, permet ainsi de tenter des extrapolations sur l'intention de l'utilisateur.

5.3.1 *Difficultés d'obtention de l'intention de l'utilisateur*

Obtenir l'intention de l'utilisateur est à la fois primordial et extrêmement difficile sans dialogue direct avec lui. Il faut donc implémenter une interface de discussion pour qu'il puisse communiquer directement des informations pertinentes. Actuellement c'est plutôt le concepteur qui peut communiquer dynamiquement en écrivant des CGs. Par la suite, le traitement ontologique d'une entrée utilisateur (une commande vocale, le traitement syntaxique d'une phrase classique) pour la transformer en CG permettra une source d'information directe et fiable. Néanmoins en l'état et pour permettre des réactions automatiques, il convient d'essayer de récupérer autant d'indices possibles permettant d'estimer cette intention. Le moteur via la sémantique des CGs et la notion de confiance peut alors fusionner ces informations et obtenir une estimation raffinée de l'intention. La généralité de l'estimation sera d'autant meilleure que la quantité d'informations fusionnée sera importante.

Ainsi des indices sont à extraire du comportement vis-à-vis de deux dimensions récurrentes des applications : l'espace et le temps. L'utilisateur peut se focaliser sur une zone précise de l'espace de nombreuses façons. La direction du regard, l'espace de travail, la zone accessible à manipulation, un déplacement vers ou autour d'une position de l'espace sont autant d'indices importants (section 5.3.2). De même dans le temps, une fréquence, une durée d'interaction (avec une entité ou une zone) sont également révélateurs (section 5.3.3). Enfin, des informations mélangeant les deux concepts comme la vitesse de déplacement (ou l'absence de déplacement) ou plus généralement la classification du mouvement de l'utilisateur participent également à construire une bonne estimation des intérêts (section 5.3.4).

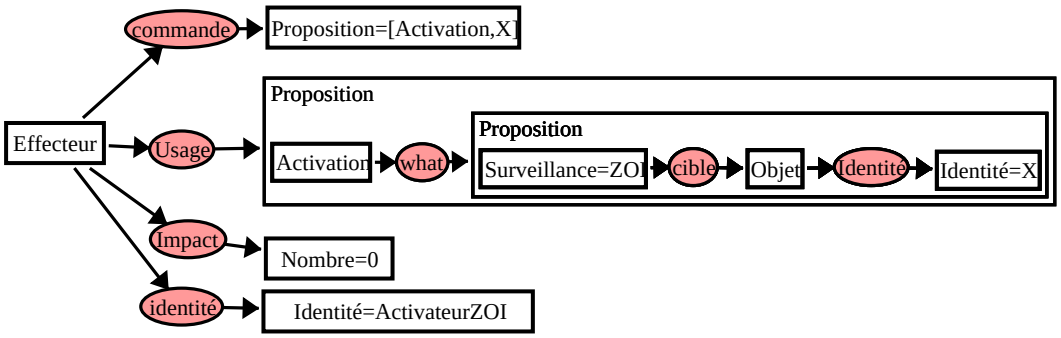
5.3.2 *Indices d'intention liés à l'espace*

5.3.2.1 *Zone d'intérêt*

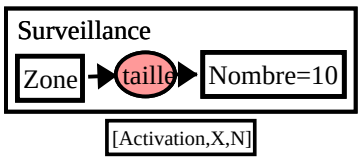
Un outil réutilisable est réalisé sous Virtools pour sonder l'espace. C'est un capteur semi-contrôlé qui permet l'*activation* de zones à créer, sensibles à la collision avec des groupes d'objets spécifiés par l'application. Cet outil gère la construction de nombreuses zones différentes. Ainsi :

- L'entrée la plus courante est l'identité d'un objet accompagnée potentiellement d'un nombre réel. La zone créée est attachée à l'objet et possède une forme identique, agrandie par défaut (d'un facteur 5) ou selon le réel. Cela permet de créer une aura, un fantôme agrandi, autour de cet objet .
- Il est possible de préciser également la forme de la zone parmi un nombre de primitives géométriques (cône, sphère, rectangle). Également en option le vecteur de position (de décalage par rapport à l'objet), de taille (avec déformation éventuelle) et de direction (par rapport à l'objet). Cela permet par exemple de créer une aura de la forme voulue, ou un faisceau dans la direction voulue, toujours attachée à l'objet.
- Enfin le type d'information qui sera précisé vis-à-vis des *objets* ou *groupes d'objets* pénétrant dans la zone peut être indiqué. Par défaut la précision effectuée est celle d'un *intérêt* via la relation *estUn*.

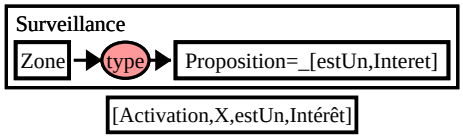
Ces fonctionnalités sont décrites comme l'*activation* de *surveillances* liées à un *objet*. Elles fournissent en effet des informations automatiquement, jusqu'à leur désactivation. La description de l'outil par défaut est représentée Fig. 90a. Il est donc possible de préciser le facteur d'agrandissement (Fig. 90b) ou les paramètres de la zone décrite (Fig. 90d). Il est également possible d'ajouter à ces trois descriptions la relation et le concept précisés (Fig 90c). Le module Virtools retourne ces informations ajoutés dans un CG à l'*objet* ou au *groupe* d'*objet* pénétrant la zone construite (Fig. 90e). Différentes descriptions de l'outil peuvent être inscrites et plusieurs usages peuvent alors être vrais simultanément. Mais seuls ceux complètement unifiés correspondent à un usage d'*effecteur* valide pour le moteur. Le module sous Virtools fait la distinction entre les différentes commandes en fonction de leur nombre d'arguments. Le moteur dispose ainsi d'un outil général pour sonder l'espace selon la situation : il peut n'avoir connaissance que d'un objet cible ou être capable de construire une zone précise. Il peut le faire pour différents types d'informations récoltées, dont les intérêts potentiels.



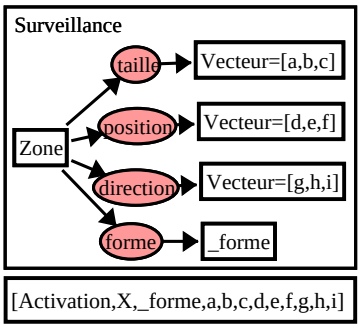
(a) Une des descriptions sémantiques de l'outil



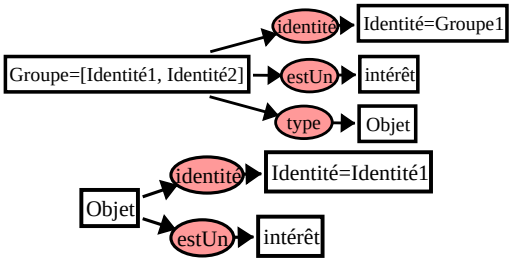
(b) Détails de la surveillance et de la commande pour une première option



(c) Détails de la surveillance et de la commande pour une deuxième option



(d) Détails de la surveillance et de la commande pour une troisième option



(e) Possibilités de CG retournés

FIGURE 90: Exemples de descriptions, variations et informations retournées par l'outil gérant les zones d'intérêts

La gestion centralisée de ces créations et activations de zones permet d'avoir un outil réutilisable du point de vue de Virtools également. Il effectue une copie dynamique de script, paramétrée pour les objets ciblés. La désactivation centralisée est alors en revanche plus compliquée à gérer. Le moteur retient automatiquement l'usage qui a été activé et l'utilise pour gérer les désactivations. Ainsi, il est plus simple pour le moteur que l'outil soit capable de retrouver la zone spécifique à désactiver à partir des mêmes paramètres que ceux correspondant à l'activation. Le nom donné à la zone est ainsi fonction des paramètres l'ayant créée. Une gestion autre est possible, en transmettant le nom de la zone au moteur, mais cela nécessite une règle liant les formes de désactivations. Étant très spécifique, elle peut être introduite dans ce cas par l'*outil* lui-même, en plus de sa description sémantique.

5.3.2.2 *Fly over et la détection d'intention*

Fly-over [11] est une technique d'interaction qui introduit une couche supplémentaire d'interprétations de la position 3D d'un pointeur réel (souris, flystick etc.) pour en déduire la position du pointeur virtuel. L'interprétation est choisie parmi trois prédéfinies, afin de s'adapter aux besoins de la tâche en cours (Fig. 91). Une

zone de repos centrale permet de travailler ou d'observer sans modification de la caméra. Une deuxième zone favorise la rotation avec une faible translation pour des déplacements précis, pour la sélection. Enfin la zone la plus éloignée favorise la translation pour un déplacement rapide sur de plus grandes distances, pour la navigation. Entre les deux se trouve une zone de transition. Cette technique permet non seulement la gestion de formes de déplacements à partir d'une seule position 3D, mais également la détection automatique de la tâche en cours (sélection, navigation ou manipulation).

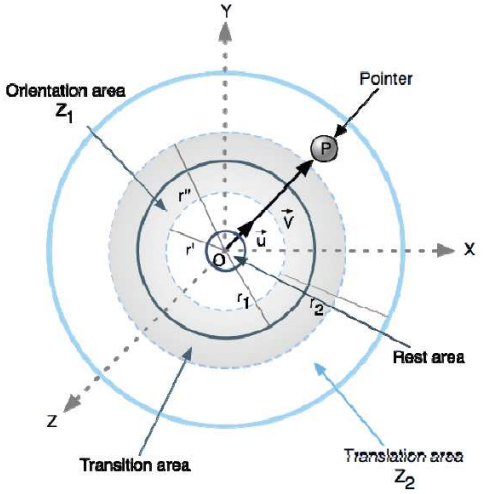


FIGURE 91: Principe de la technique d'interaction Fly over [11]

Cependant fournir au moteur l'information détectée de tâche, sans pour autant déclencher le fonctionnement correspondant sous Fly-over, laisse l'opportunité au moteur de fusionner l'ensemble des informations disponibles. Ainsi un contexte autre que spatial [3] peut être agrégé pour déterminer la tâche la plus probable. Désolidariser la détection de l'application de la tâche dans Fly-over est ainsi une piste intéressante. Mais pour profiter au mieux des deux aspects (intelligence embarquée et centralisée), une approche est de permettre au moteur de décider de la tâche en cours, en ayant la main mise sur l'étage final de Fly-over. Fly-over fonctionne déjà à l'aide de messages entre ses différentes parties. Un *outil* est donc créé qui gère les commandes issues du moteur et les rend prioritaires sur les messages internes de Fly-over. Celui-ci devient donc également un effecteur permettant de sélectionner la tâche en cours, en plus d'une détection de la tâche probable initiale. Son fonctionnement d'interaction 3D adaptative indépendant est conservé mais il peut également être contrôlé par le moteur.

Enfin Fly-over permet également une sélection des objets présents sous le pointeur 3D via un 2D picking. Le procédé est le même que pour la sélection à l'aide d'un pointeur de souris, à l'exception du fait que le pointeur 3D peut être placé derrière un premier objet pour sélectionner quelque chose en arrière plan. Cette détection est également transformée en capteur d'intérêt pour le moteur. Un chronomètre évite de rajouter un objet dès qu'il est survolé. A partir d'une demi-seconde l'intérêt est reporté avec une faible confiance (0.1). Par la suite toutes les secondes où le pointeur reste sur cet objet, ce même fait est à nouveau ajouté dans la base de

3. Fly-over utilise en pratique également le temps pour effectuer des transitions souples entre les zones et éviter les changements inopinés.

connaissance. La gestion automatique de la fusion d'information par le moteur permet donc un renforcement de la confiance en cet intérêt, basé sur ces entrées multiples.

5.3.3 *Indices d'intention lié au temps*

Les intérêts liés au pointeur de Fly-over présentés ci-dessus intègrent une gestion du temps. Il est possible de réaliser d'autres mesures sous Virtools détectant de nouveaux intérêts liés au temps. Par exemple détecter un nouvel intérêt si l'utilisateur reste proche, interagit longtemps ou fréquemment avec un objet seraient autant de nouveaux capteurs potentiels. Cependant une première approche a été d'exploiter les intérêts déjà déclarés par l'application (par quelque moyen que ce soit, notamment via la gestion de l'espace) et de les raffiner en gérant le temps. Ainsi l'obtention d'intérêts liés au temps se fait ici sans capteurs supplémentaires, mais via la gestion de règles et de concepts au sein du moteur.

Un *intérêt* passé mais encore présent dans l'*historique* peut être considéré comme un possible *intérêt* actuel (Fig. 92a). Cette règle n'étant pas vraie dans la majorité des cas, sa confiance est faible. La confiance élémentaire maximale générée par cette règle est égale au dixième de la confiance précédente en cet intérêt avant son passage dans l'historique. Elle permet ainsi potentiellement d'obtenir de nouveaux intérêts accessibles par la base de connaissance.

Un intérêt durable est plus sûr. Ainsi un chemin supplémentaire est introduit [4] fonction de la confiance en cette *durée élevée* (Fig. 92b). Cette règle possède une confiance plus élevée car elle n'introduit pas de nouvel intérêt et est globalement plus souvent vraie. Cela ne concerne donc que les intérêts qui possèdent une date (à partir de laquelle la durée est définie). Ainsi deux comportements s'opposent ici : la péremption menant à une confiance déclinante des évènements et ce renforcement pour les intérêts. Cela ralentit en pratique la perte de confiance des intérêts mais bénéficie directement à ceux dont les évènements déclarants se chevauchent [5].

La fréquence et la durée élevées d'un intérêt ont toutes deux un effet positif sur sa confiance. La gestion interne de ces raisonnements permet d'en bénéficier quelle que soit la source d'information déclarant des intérêts.

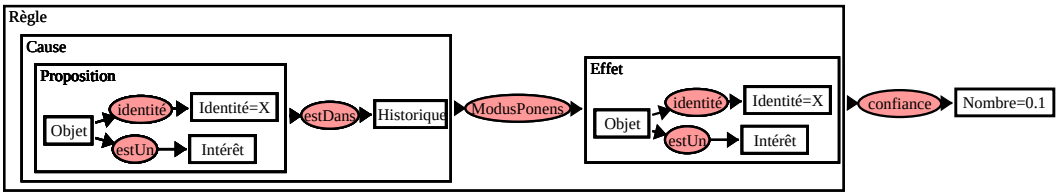
5.3.4 *Indices d'intention liés au mouvement*

5.3.4.1 *Attributs du mouvement*

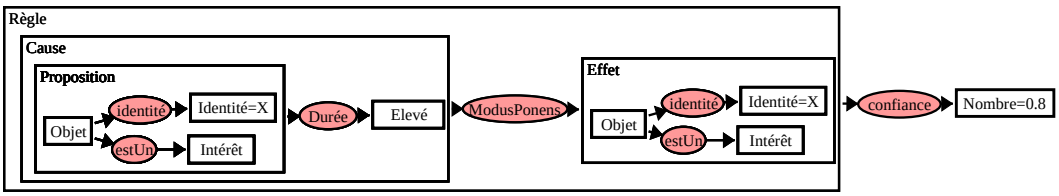
A cheval entre les intérêts liés au temps et à l'espace, sont ceux définis par le mouvement. Ainsi si un utilisateur reste immobile près d'un objet, il est possible qu'il ait un intérêt particulier pour celui-ci. Une autre explication serait qu'il soit dans une tâche d'observation de l'environnement. Il faut donc prendre en compte un immobilisme relatif uniquement si l'objet proche est déjà un intérêt ou intro-

4. Vouloir renforcer un intérêt, en utilisant directement ce même intérêt sans précaution, peut déclencher une boucle infinie : l'effet est alors une spécialisation de la cause.

5. Ce qui amène à une gestion complexe d'un *quantitatif* pour la *durée* qui est étendue à l'*historique*. De plus, l'extension doit s'arrêter dans le cas d'une information contraire certaine. Par exemple une succession d'états actifs interrompue par un état inactif explicite.



(a) Persistance partielle des intérêts passés



(b) Augmentation de la confiance des intérêts prolongés

FIGURE 92: Les règles obtenant des intérêts liés au temps

duire ce nouvel intérêt avec une confiance faible. Le premier cas correspond à une gestion par le moteur de l'information supplémentaire pour renforcer un intérêt, le deuxième cas à une source d'intérêts issus de l'application. Dans les deux cas, il faut donc un capteur pour pouvoir relever un *mouvement* que nous qualifierons de *local* (un immobilisme complet est rarement atteignable). Au contraire lorsque le mouvement est *global*, et possède notamment une direction privilégiée, une hypothèse probable est qu'il se trouve un intérêt particulier dans cette direction. Dans ce cas, sonder l'espace dans cette direction peut être souhaitable. Mais, à nouveau l'utilisateur peut être en train d'explorer l'environnement. Partant de l'hypothèse que le cas de l'exploration serait également celui d'un mouvement plus hésitant, il est utile de qualifier également la vitesse du mouvement pour aider à distinguer les deux cas. Ainsi un mouvement *global* et *élevé* peut devenir une source d'intérêt. Le capteur permettant de caractériser le type de *mouvement* de l'utilisateur, *local* ou *global*, à vitesse *faible* ou *élevée* peut être ainsi utilisé de façons multiples. Il peut permettre également de repérer d'éventuelles difficultés de l'utilisateur (section 5.4).

L'obtention de ces informations se fait par l'implémentation d'un capteur semi-contrôlé qui permet le déclenchement d'une *surveillance* du *mouvement* d'un *objet* (l'outil est décrit de manière similaire à Fig. 90a). Pour une cible correspondant à l'*utilisateur*, comme la *main* ou l'*avatar* , nous obtenons les comportements présentés précédemment. Mais l'étude du mouvement d'un autre objet donné peut être également source d'informations. Ces informations sont obtenues en comparant le ratio de la valeur absolue de la vitesse moyenne et la vitesse absolue moyenne de l'objet. Plus ce ratio est élevé, plus la vitesse moyenne est proche de la vitesse absolue moyenne (Fig. 93a). Dans ce cas la vitesse possède une direction privilégiée et la trajectoire possède alors une logique globale (Fig. 93b). Un ratio faible implique un mouvement local (Fig. 93c). Pour savoir si le mouvement est élevé ou non, il suffit d'effectuer un second ratio, cette fois entre la vitesse absolue et un ordre de grandeur de la vitesse maximale (ce ratio possède une saturation supérieure et reste inférieur à 1).

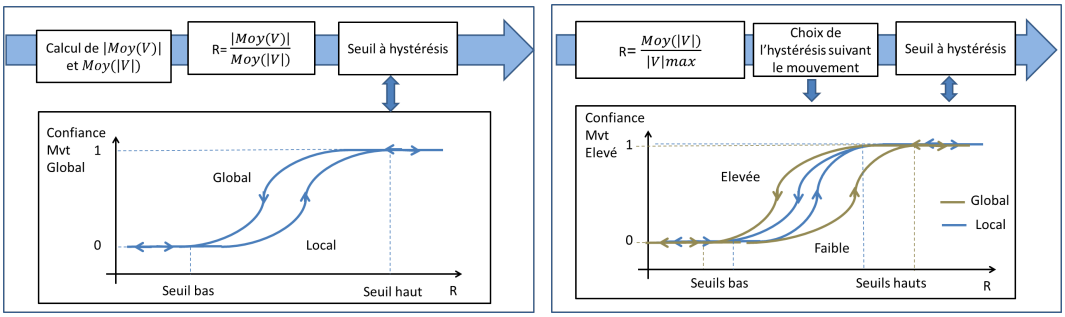
$Moy(|V|) = |Moy(V)|$ (a) Mouvement global idéal

$Moy(|V|) \gtrsim |Moy(V)|$ (b) Mouvement global

$Moy(|V|) \gg |Moy(V)|$ (c) Mouvement local

FIGURE 93: Types de mouvement et moyennes associées

Dans les deux cas (Fig. 94), on transforme la valeur du ratio continue obtenue avec une fonction non linéaire pour représenter qualitativement ces concepts (avec le même raisonnement que pour les attributs). En revanche est ajouté un cycle d'hystérésis pour pallier les fluctuations du calcul de la moyenne. Un mouvement qui a possédé une structure interne suffisante pour atteindre un niveau élevé de confiance dans une des qualifications du mouvement peut alors conserver cet état plus facilement. Un seuil haut sur le ratio permet ainsi à un mouvement très probablement global de conserver plus longtemps cette confiance lorsque le ratio diminue (et réciproquement pour un mouvement local, Fig. 94a). De plus nous n'avons pas la même impression qualitative des concepts *faibles* et *élevés* pour un mouvement *local* ou *global*. L'hystérésis associé au choix de ces concepts est dépendant du résultat sur la portée du mouvement (Fig. 94b). Il faut donc obtenir une vitesse plus importante pour qualifier un mouvement *global* comme étant *élevé*. De plus, les mouvements locaux pouvant être plus chaotiques, son hystérésis plus fin avec des zones de saturation plus longues filtre mieux les fluctuations.



(a) Qualification de la zone de mouvement (b) Qualification de la vitesse de mouvement

FIGURE 94: Processus du capteur d'attributs du mouvement

5.3.4.2 *Classification de trajectoire*

Toujours afin d'avoir des outils généraux dans leur implémentation, il serait utile de pouvoir classifier des données sans créer un outil spécifique pour chaque cas. Une méthode par apprentissage est donc intéressante. Ainsi dans le cadre d'une trajectoire d'objets, notamment pour celle liée à la main menant à une reconnaissance de gestes, un module basé sur les HMM discrets a été réalisé [34]. Néanmoins ce même module peut en fait être utilisé pour la reconnaissance de n'importe quelle série (temporelle) de données (échantillonnées). De plus, les données peuvent être pré-traitées avant la reconnaissance elle-même, ce qui permet via l'usage de cette

fonction de codage au niveau de ce capteur de gérer différents formats de données, de les fusionner au besoin ou d'obtenir plusieurs types de reconnaissance à partir de données similaires. Par la suite, le module se focalise sur la reconnaissance de gestes à partir de positions 3D de la main, information disponible par la grande majorité des moyens d'interaction.

La détection se base donc sur l'observation de séries de symboles (figure 95), sélectionnées à partir d'un flux continu de la mesure. Ces séries peuvent être de longueurs différentes selon le geste et mutuellement inclusives (car elles correspondent à des durées plus ou moins longues situées autour d'un même instant).

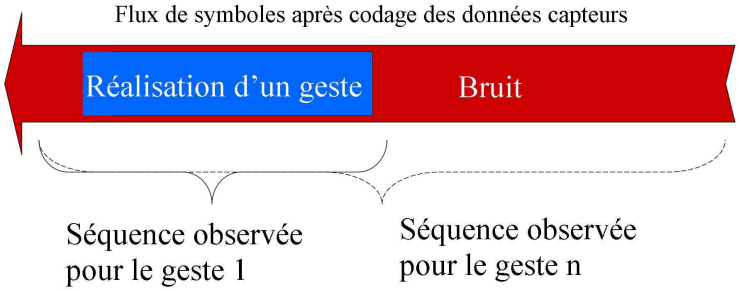


FIGURE 95: Schéma de l'observation

Pour déterminer si ces séries contiennent ou non un geste, on utilise comme élément discriminant un HMM par geste, afin d'évaluer une vraisemblance de la génération de cette séquence par cette classe (grâce au chemin d'états le plus probable). La détection est alors basée sur ce principe simple : après apprentissage, la séquence contenant un geste devrait rendre maximale la vraisemblance du HMM associé. Il reste alors à comparer la vraisemblance des gestes à un seuil pour valider la détection. Un des intérêts majeurs de cette approche est de pouvoir reconnaître un geste même si celui-ci connait des variations. Notamment cela permet d'éviter de signaler au système le début d'un geste. En effet le score du HMM associé devrait être plus important que celui du seuil car il reste le plus apte à générer la plus grande part du signal. On pressent déjà que pour de bonnes performances il faudra utiliser un seuil intelligent [53], que l'on gérera finalement aussi par un HMM, entrainé spécialement. On obtient ainsi l'algorithme présenté Fig.96.

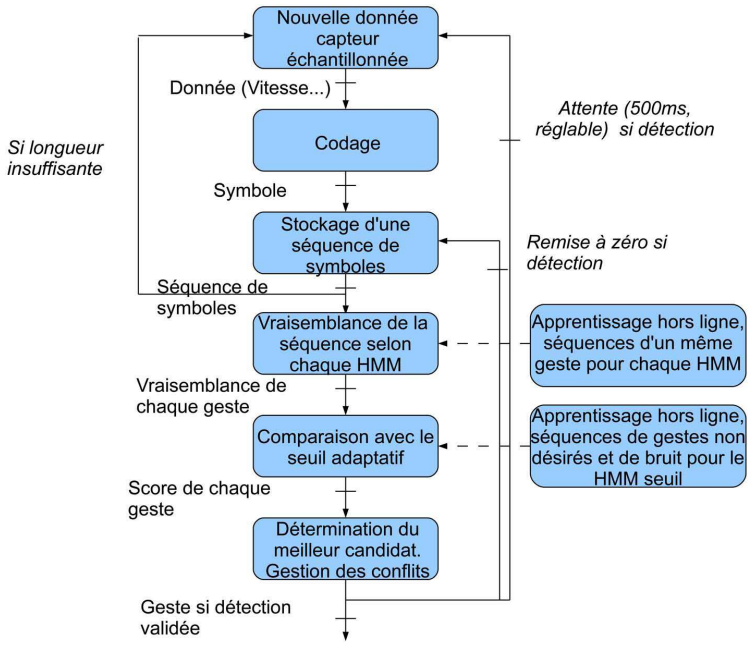


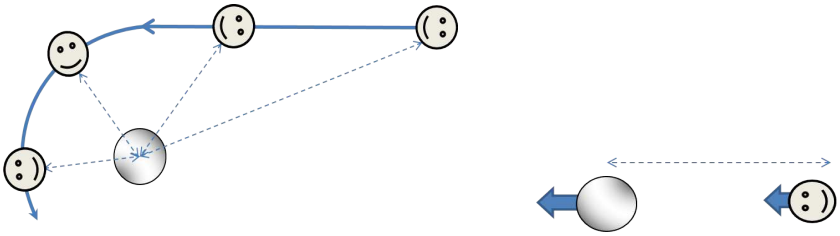
FIGURE 96: Chaîne de traitement pour la reconnaissance de gestes

5.3.4.3 *Capteurs envisagés pour les mouvements relatifs*

L'étude de l'évolution du mouvement relatif de l'utilisateur par rapport à un objet permet également de déterminer ou de qualifier un intérêt. Premièrement, si l'intérêt est connu, l'étude de ce mouvement permet d'avoir des indices sur l'objectif (une observation ou une sélection). Il est également possible de déterminer de nouveaux intérêts via l'étude de l'ensemble des mouvements relatifs par rapport à l'ensemble des objets potentiels d'intérêts (un groupe d'objet manipulable sous Virtools utilisé également par le capteur de zones d'intérêts). Par exemple, dans le cas où l'utilisateur se rapproche qualitativement d'un ou de plusieurs objets en particulier, ceux-ci peuvent être relevés comme intérêts. Un script peut être donné à chaque objet potentiel d'intérêt sous Virtool afin de détecter une telle approche de l'utilisateur (via un seuil, un hystéris, le classifieur HMM précédent, etc.). Cela se rapproche de l'étude de mouvement global de l'utilisateur avec chaque objet pouvant participer à l'étude.

Mais d'autres cas peuvent être résolus par l'étude d'un mouvement relatif. Par exemple, celui où l'utilisateur garde une distance globalement constante par rapport à un objet en particulier, alors même que l'utilisateur est en mouvement. En effet, un utilisateur effectuant un déplacement entre deux points peut avoir son attention attirée par une zone ou un objet particulier. Tout en continuant son déplacement initial dans une direction donnée, son regard et souvent sa trajectoire s'incurvent vers cet intérêt (Fig. 97a). Cette forme de trajectoire, de type gravitation, peut être détectée pour reporter son centre d'attraction comme zone ou objet d'intérêt. Un deuxième moyen de la qualifier serait l'apparition d'un palier de distance relativement constant par rapport à cet objet. Enfin, un autre indice peut être l'évolution de l'angle entre la direction du déplacement et le regard, si ce dernier est suivi. L'apparition de paliers constants successifs peut aussi se détecter lors de la poursuite d'un objet mobile (Fig. 97b). La détection de cette situation spécifique permet potentiellement de résoudre via le moteur un défaut de Fly-over. Celui-ci

nécessite un arrêt partiel avant de pouvoir effectuer une sélection. Un objet en mouvement peut alors distancer l'utilisateur pendant cette transition.



(a) Intérêt gravitationnel (b) Intérêt poursuivi

FIGURE 97: Étude de mouvement relatif

5.3.5 *Obtention d'objectifs*

Au delà des intérêts, des objectifs peuvent être définis. Définir les objectifs sans commande directe de l'utilisateur est complexe et d'autant plus risqué que les adaptations alors débloquées ont potentiellement plus d'impact. Ainsi l'application peut proposer à l'utilisateur des commandes spécifiques comme pouvoir pointer un objectif de la main. En combinant la reconnaissance de gestes pour des mouvements de la main rectilignes et la création de zones d'intérêts, l'application peut alors définir un tube dans la direction du geste. Cela peut se faire au sein du moteur par la définition de règles spécifiques gérant le lien entre les différent outils (la reconnaissance de gestes, l'obtention d'attributs des gestes et la création de zone). Ainsi lorsqu'un geste rectiligne est déclenché, une zone d'intérêt directive est créée à l'issue de la question permettant d'obtenir la direction d'un geste (Fig. 98). La combinaison de ces outils peut être directement effectuée dans l'application pour obtenir un capteur non contrôlé d'objectif.

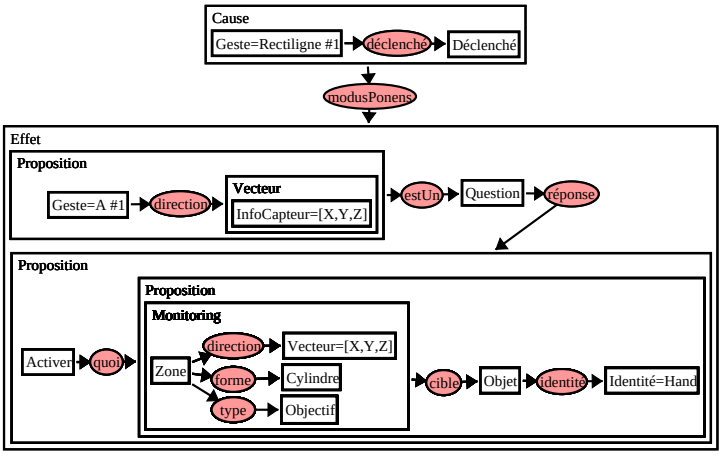
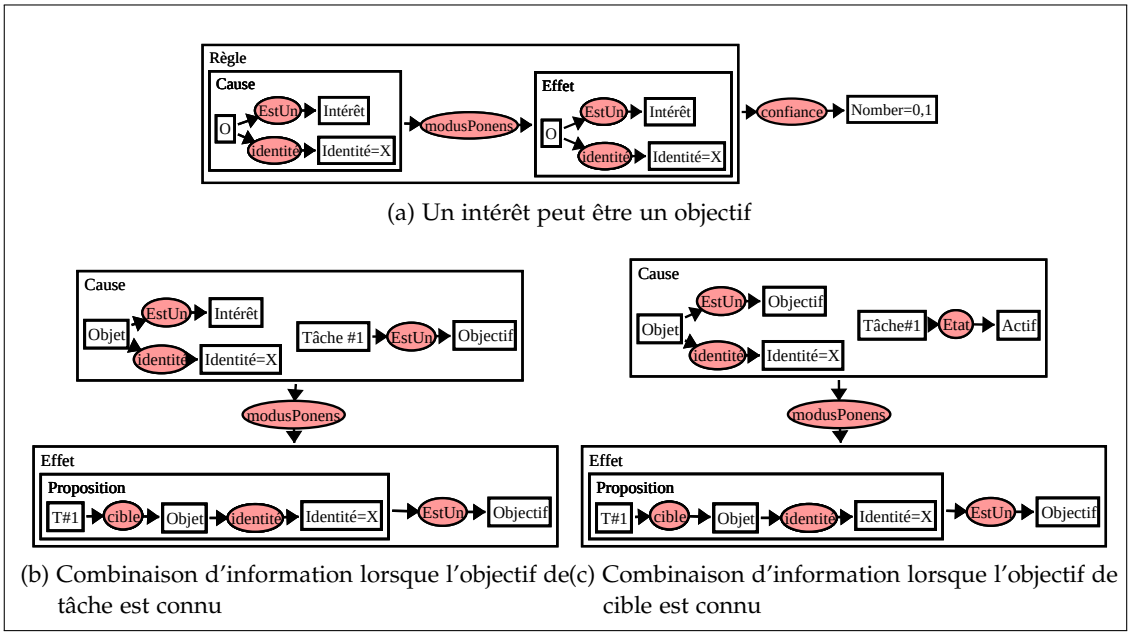


FIGURE 98: Règle d'obtention d'objectif pointé

Enfin, l'application peut parfois connaitre l'objectif final ou temporaire de l'utilisateur (par exemple dans les scénarios d'apprentissage virtuel). La combinaison d'informations permet également d'estimer des objectifs (Fig. 99). On peut ainsi supposer avec une faible confiance qu'un intérêt est également un objectif (Fig. 99a). Autrement, connaissant une partie de l'objectif, des règles tentent sa complétion sans modifier la confiance des sources d'information (Fig. 99b et 99c). Il est intéressant par la suite de pouvoir définir comme objectif l'exploration libre, ce qui est

différent de l'absence d'objectif. En effet cette dernière peut être due à l'absence de capteurs adaptés.



(a) Un intérêt peut être un objectif

(b) Combinaison d'information lorsque l'objectif de tâche est connu

(c) Combinaison d'information lorsque l'objectif de cible est connu

FIGURE 99: Les règles obtenant des objectifs

5.4 DÉFINITION D'ÉTAPES DANS LE RAISONNEMENT

Afin de rendre le raisonnement tant que possible indépendant à la fois des capteurs et des effecteurs disponibles, introduire des abstractions supplémentaires comme étapes de raisonnement est utile. Elles permettent à la fois d'isoler le raisonnement, d'éviter une description trop spécifique mais également de fusionner les informations à chacune de ces étapes. Ces combinaisons d'éléments de contexte pour définir des situations types sont à mi-chemin entre l'obtention de contexte et l'adaptation.

5.4.1 *Types d'étapes*

Des étapes de raisonnement ont déjà été rencontrées. La section précédente présente des capteurs externes ainsi que des règles internes pour obtenir ou renforcer des intérêts et objectifs. Ces intentions estimées sont essentielles pour la poursuite du raisonnement et en sont une des premières étapes. Des concepts comme les états disponibles ou actifs, des jugements de faits binaires (section 4.5.2), permettent à leur tour souvent l'expression d'étapes. Afin par exemple de préciser des propriétés à ne consulter uniquement que si la ressource correspondante est disponible ou si l'état est actif. Enfin, leur changement de valeur n'est pas direct mais se fait à travers des volontés gérées en interne[6]. Elles aussi sont des étapes du raisonnement.

Ainsi parmi les étapes de raisonnement se trouvent deux grand genres :
– l'expression de types de situations récurrentes, associées à des états particuliers qui peuvent être propices ou non à l'assistance.

6. La volonté d'activer mène à une possible activation et à un état actif. Une disponibilité variable de ressources est prévue.

– des volontés d'actions qui permettent d'exprimer des types d'actions, suivis d'effets différents selon le reste du contexte. Elles servent entre autres de transitions entre états[7]. Mais également à exprimer des types d'assistances qui peuvent être interprétées et/ou réifiées de plusieurs manières.

Ces étapes représentent donc globalement des situations promptes à déclencher des réactions. Elles sont de plus en plus inter-connectées et pas toujours aisées à présenter de manière linéaire. Dans cette section, la volonté de *valoriser* est introduite (section 5.4.2). Celle-ci n'est à effectuer que si elle n'est pas déjà accomplie. L'état *accompli* permet d'exprimer généralement des finalisations (section 5.4.3). Puis est discutée la volonté d'*utiliser* une ressource, notamment les techniques d'interactions selon la situation (section 5.4.4). La technique courante peut donc être plus ou moins adéquate. La situation peut en devenir plus ou moins difficiles. Des jugements de valeur sur la situation qui peut être *ordinaire* ou non, *facile* ou non, sont alors introduits (section 5.4.5). Selon ces difficultés, il est envisageable de vouloir *modifier* la situation, comme les techniques d'interaction courantes (section 5.4.6).

5.4.2 *La volonté de valoriser les objets*

Une des adaptations simples et réactives à mettre en place est la mise en valeur d'objets. Ces valorisations peuvent être déclenchées en fonction des attributs pour un objet détecté au préalable comme un intérêt. Par exemple, elles sont utiles afin d'aider à repérer un objet pour une navigation ou une sélection. Ainsi quelles que soient au final les adaptations déclenchées pour y répondre, la règle Fig. 107 statue sur la volonté de *valoriser* même. La *valorisation* n'est pas nécessaire si elle est déjà accomplie, peut-être de manière implicite lorsque l'attention de l'utilisateur est clairement portée sur l'objet. Ainsi si l'objet est manipulé ou si l'objet fait partie d'un objectif accompli, il est inutile de le valoriser. La volonté de valoriser ou la valorisation même peuvent également être indiquées directement par l'application selon son scénario ou par l'utilisateur selon son envie. L'ensemble est, comme pour toute autre information, agrégé pour déterminer la confiance globale.

7. États et volontés d'actions peuvent ainsi servir à créer des sortes de machines d'états. Mais à moins de se restreindre à des causes binaires, les états peuvent être simultanément vrais à des degrés différents et partagés complètement ou partiellement par plusieurs machines d'états.

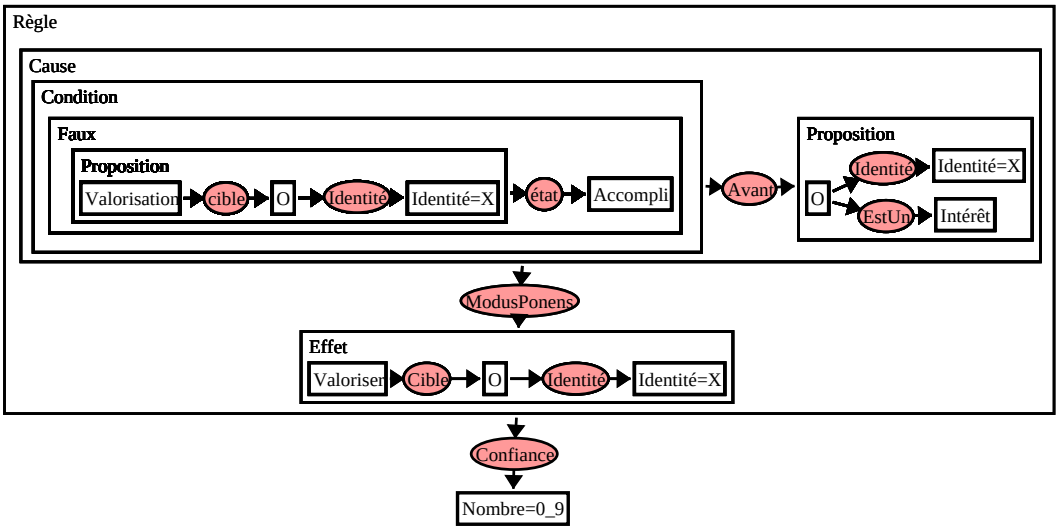


FIGURE 100: Volonté de valorisation des intérêts

Introduire l'étape intermédiaire de valorisation accomplie permet de conserver cette règle quelles que soient les définitions de cette accomplissement (autrement l'ajout d'une définition ajouterait une cause). A noter que l'usage du faux est ici utile : si l'information d'accomplissement ne peut être obtenue, elle n'est pas bloquante pour la volonté de valorisation. Enfin, une règle similaire de confiance unitaire exprime la volonté de *valoriser* un objet qui est un *objectif*.

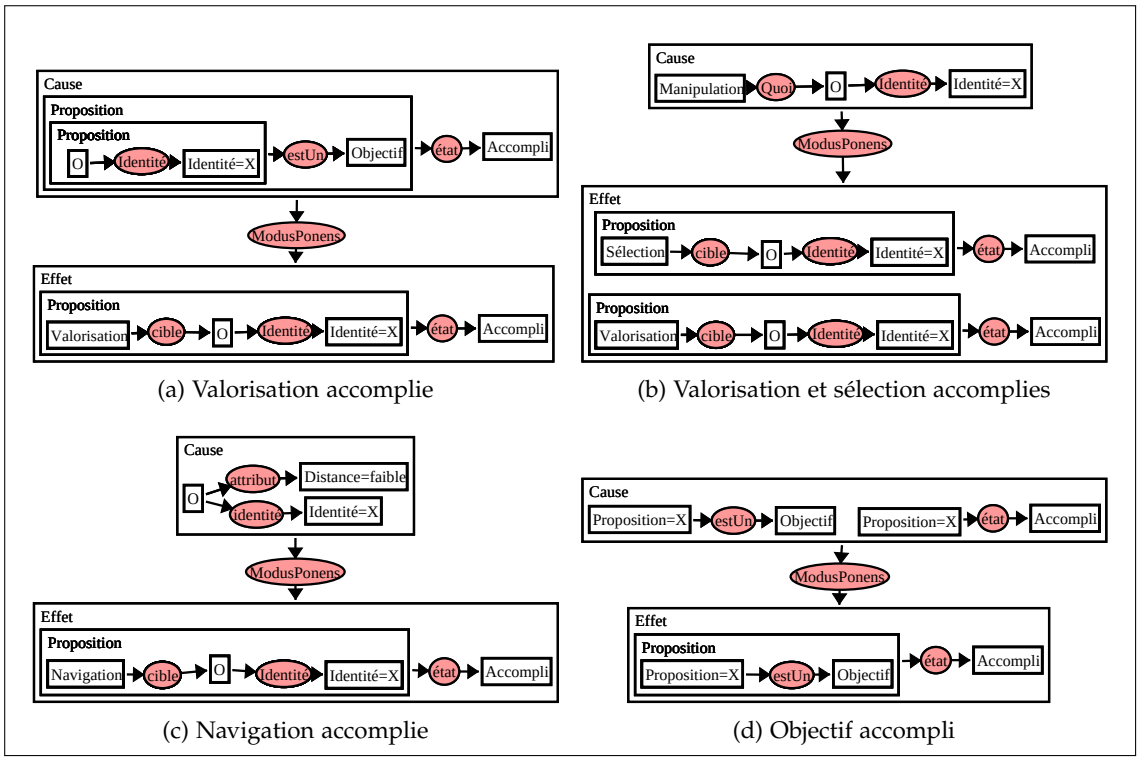
5.4.3 *Notion d'état accompli*

La notion d'état *accompli* peut être utilisée comme un jugement de fait. Il peut être indiqué par l'application ou l'utilisateur. Mais souvent il faut estimer cette notion pour un objectif ou une tâche sans pour autant avoir de certitudes. C'est donc en pratique un jugement de valeur. Dans les deux cas, cette notion exprime l'idée de finalisation. Celle-ci peut n'être que temporaire et correspond bien à un état (un objectif déjà accompli peut être à nouveau visé). Si un parallèle peut être fait entre état *actif* et *inaccompli*, les deux notions n'expriment pas les mêmes choses. *Inaccompli* sous-entend que l'accomplissement, la finalisation est recherchée. *Actif* est l'indication d'une situation ayant un effet actuel.

Un exemple pour la valorisation, qui peut-être ou non accomplie, a été introduit précédemment. Un objet manipulé revêt déjà une importance particulière et reçoit déjà l'attention de l'utilisateur (Fig. 101b). Une aide peut être désirée mais celle-ci ne revêt alors plus la forme d'une *valorisation* (que celle-ci aie d'ailleurs été effectivement *active* ou non). De même, si l'objet a eu plus généralement part à un autre objectif *accompli* (e.g. la navigation vers cet objet), la *valorisation*, ayant pu aider à cet accomplissement, l'est alors également (Fig. 101a).

Enfin, en combinant la notion d'objectif avec d'autres éléments de contexte, le moteur peut déterminer la confiance qu'une tâche soit accomplie. La sélection d'un objet manipulé est accomplie (Fig. 101b). De même, la navigation vers un objet qui est qualifié comme proche est vraisemblablement accomplie (Fig. 101c). Enfin si un objectif est inclus dans une description d'*état* accompli, il peut être lui-même marqué comme accompli (Fig. 101d). Un objectif accompli permet alors de déclen-

cher d'autres règles (e.g l'inscription de cette étape dans l'historique, désactiver des aides actives le concernant etc.).



(a) Valorisation accomplie

(b) Valorisation et sélection accomplies

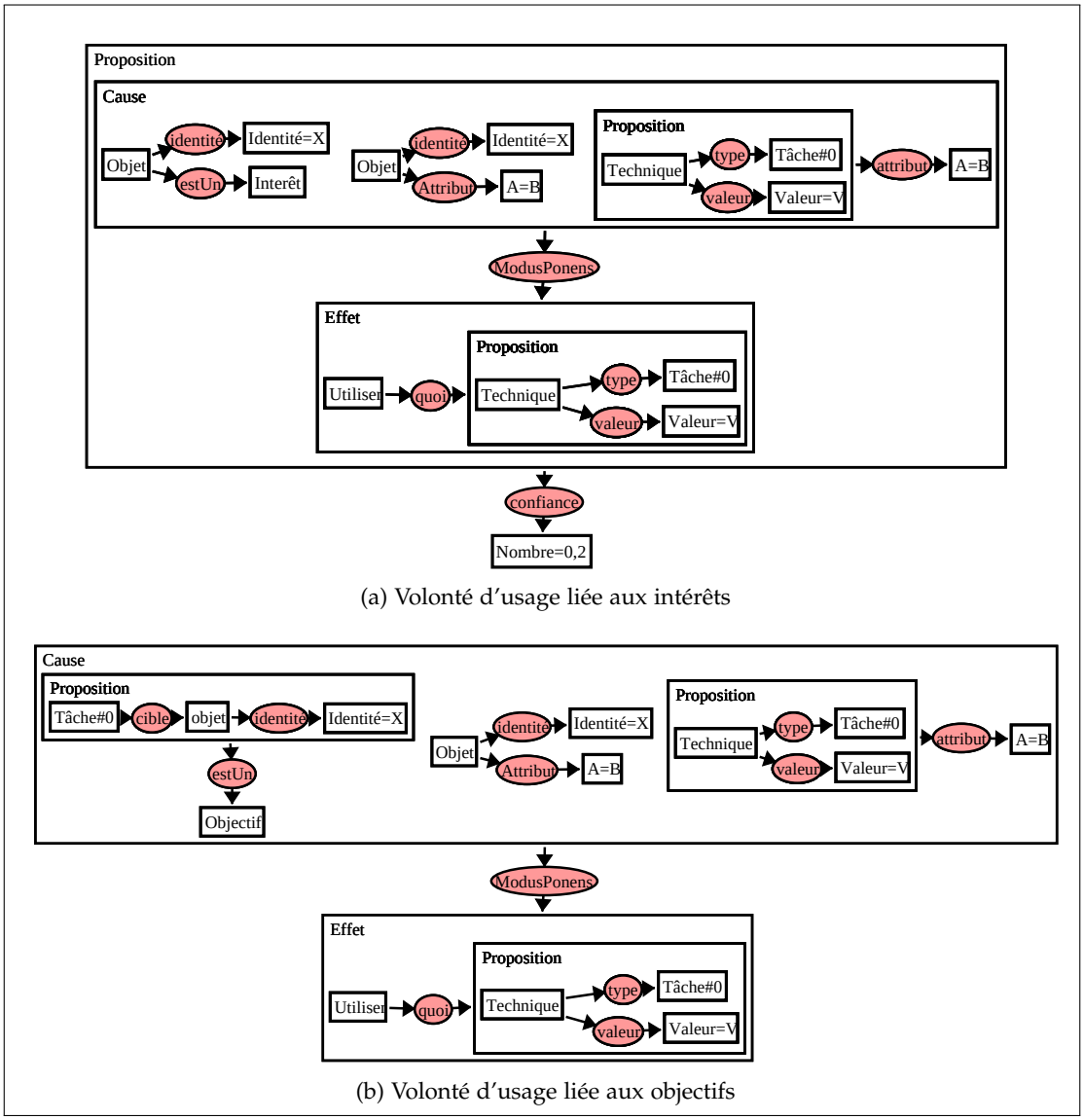
(c) Navigation accomplie

(d) Objectif accompli

FIGURE 101: Règles définissant des accomplissements

5.4.4 *La volonté d'utiliser une ressource*

L'usage d'une ressource comme une technique ou une modalité peut être plus ou moins désiré selon la situation. Par exemple, à partir de la notion d'*état disponible*, de l'ajout de propriétés a priori et de la situation courante, la volonté d'utiliser une technique d'interaction peut être différente. Si une technique d'interaction permet la bonne gestion des objets d'intérêt, la confiance de son usage, actuel ou non, est bonne (Fig. 102a). De même, dans le cas d'un objectif lié à un objet connu, l'adéquation des techniques vis-à-vis de ce dernier est importante, avec une confiance de règle plus élevée (Fig. 102b). A nouveau d'autres sources peuvent influer sur cette volonté d'usage. Si l'utilisateur préfère *utiliser* une technique particulière, il peut l'indiquer directement. Si cette préférence est conditionnelle, une règle est à ajouter. L'ensemble est alors agrégé, préférences et contexte de l'environnement, lors de la détermination de la confiance de chaque usage.



(a) Volonté d'usage liée aux intérêts

(b) Volonté d'usage liée aux objectifs

FIGURE 102: Volonté d'usage d'une ressource

5.4.5 *Notion d'états ordinaires et difficultés*

Parmi les états intéressants sont ceux permettant de porter un jugement de valeur sur la situation (résumé Fig 103). Ainsi la qualification de l'*état* d'une situation comme étant *ordinaire* ou non est une étape importante pour l'adaptation. Plusieurs degrés d'ordinaires sont possibles : habituel, normal, quelconque etc. Nous n'utilisons pour l'instant que le terme de *normal* classé donc comme un sous-type de l'*ordinaire*. *Normal* est ici utilisé pour qualifier une situation à laquelle on peut s'attendre sans en qualifier la simplicité. *Facile* et *difficile* sont des *contraires* qui permettent de qualifier ce degré.

Le *contraire* d'un *état ordinaire*, une situation *particulière*[8] est également introduit. Cet état doit pouvoir de manière privilégiée déclencher une assistance. Notamment l'état *anormal* qui est un sous-type introduit ajoutant un jugement de valeur négatif : une adaptation est alors nécessaire pour y remédier. *Anormal* et *normal* sont des

8. Le contraire utilisé n'est pas extraordinaire qui est plutôt mélioratif.

contraires [9]. Une situation peut être compliquée au sens anormalement complexe. Dès lors, si *difficile* n'est pas pour autant *anormale*, l'accumulation de difficultés peut être liée à l'apparition d'une situation anormale. Une règle peut être ajoutée pour obtenir cette propriété.

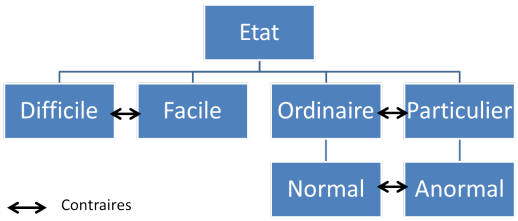
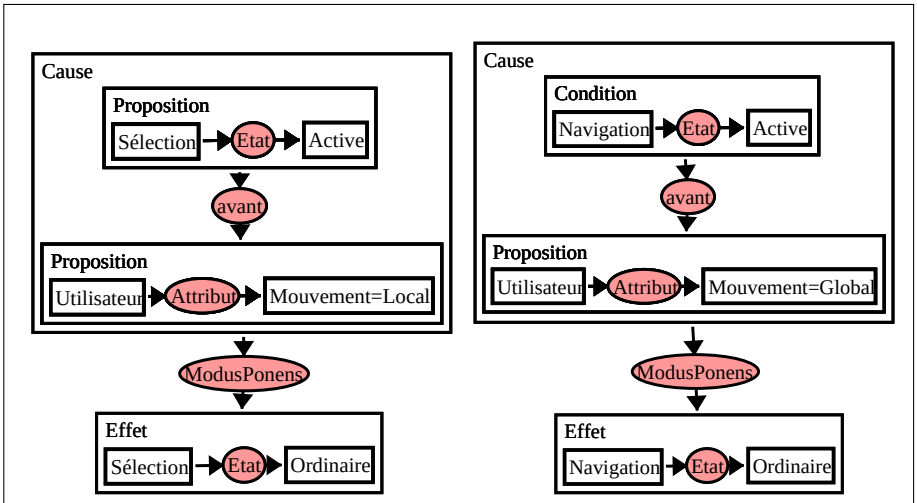


FIGURE 103: Récapitulatif des concepts utilisés pour décrire un état via un jugement de valeur

Des règles permettent de définir différentes situations ayant un état *ordinaire*. Une tâche de sélection ordinaire s'effectue avec un utilisateur au repos ou faiblement mobile (Fig. 104a). Une navigation ordinaire s'effectue avec une direction globalement définie (Fig. 104b). Le contraire de ces situations *ordinaires* ne correspond pas à des situations *anormales* mais *particulières*. Ainsi une navigation dans le cas d'une exploration ou d'une observation peut s'effectuer par des mouvements locaux également. La détection de la particularité est intéressante sans pour autant constituer un problème.



(a) Sélection ordinaire un utilisateur faiblement mobile

(b) Navigation ordinaire avec une direction privilégiée

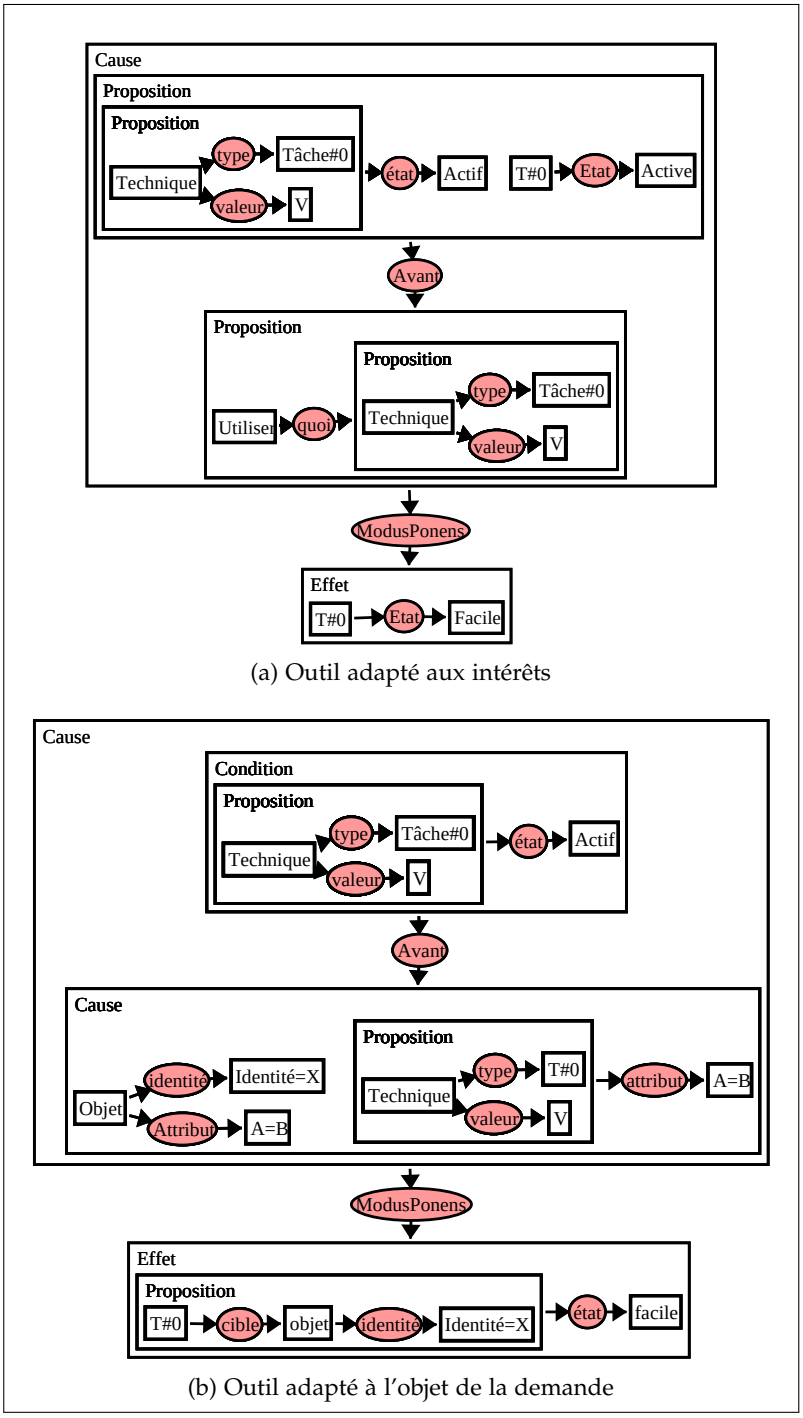
FIGURE 104: Exemple de situations ordinaires

Une *tâche* est considérée comme *facile*, si la technique correspondante *active* a un usage adapté aux objets d'intérêts (Fig. 105a). L'usage adapté est représenté par la volonté d'*utiliser* de cette technique (présentée section 5.4.4). Enfin, il est possible d'obtenir l'information de l'adéquation de la technique pour un objet spécifique (Fig. 105b). Une situation difficile n'est donc pas pour autant anormale ou particulière, mais permet de prendre des mesures adaptées. Le changement

9. Il faut le préciser pour pouvoir déduire qu'une situation qui n'est plutôt pas *normale* est non seulement plutôt *particulière* mais également plutôt *anormale*

de technique d'interaction peut ainsi être envisagée pour une interaction difficile (choix de la technique d'interaction section 5.6).

Enfin, l'apparition ou le maintien d'un état *ordinaire* peut également servir de confirmation implicite vis-à-vis de l'adéquation d'une réaction précédente. Des situations *ordinaires* ou *faciles* peuvent être jugées plus propices à la présentation d'information ou à la sollicitation de l'utilisateur et plus généralement au déclenchement de réactions à fort *impact*.

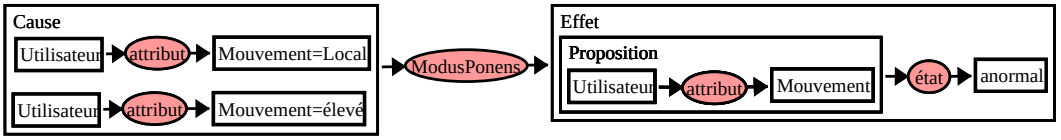


(a) Outil adapté aux intérêts

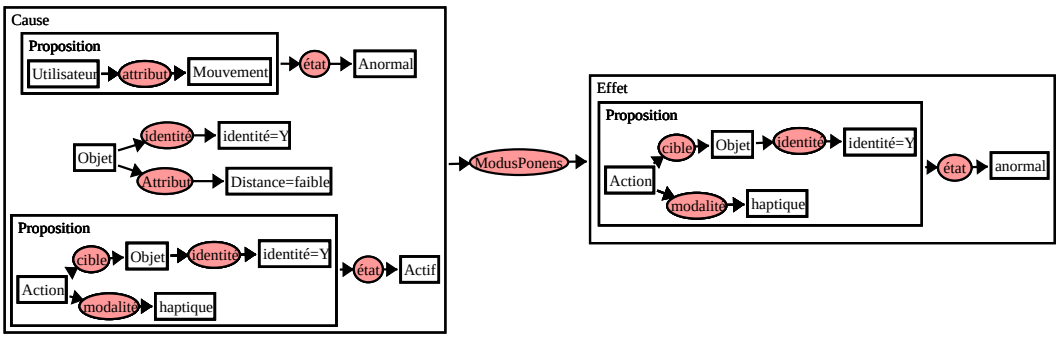
(b) Outil adapté à l'objet de la demande

FIGURE 105: Exemple de situations faciles

Lorsque l'utilisateur effectue des mouvements locaux et élevés, son mouvement est considéré anormal (Fig. 106a). L'apparition d'un état *anormal* peut également servir d'infirmation implicite à une réaction précédente. Ce type de mouvement utilisateur peut être du à une modification actuellement active de l'interaction ou à une action appliquée à l'aide d'une modalité haptique, exercée directement sur l'utilisateur ou sur un objet proche (Fig. 106b). Cette action est alors jugée anormale (et peut déclencher sa désactivation).



(a) Détermination d'un mouvement anormal



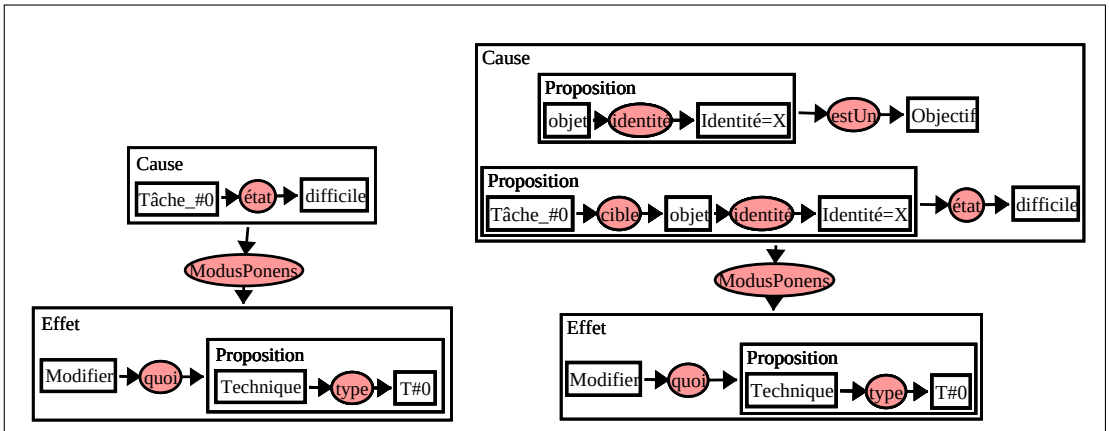
(b) Détermination d'une modalité haptique gênante

FIGURE 106: Exemple de situations anormales

Enfin l'historique peut être plus exploité pour ces notions. L'alternance de tâches rapides permet de repérer une difficulté. Une tâche de manipulation très courte peut traduire une erreur de sélection. Une alternance de tâches de sélection et de manipulation permet de détecter la difficulté de sélection du bon élément. La succession de navigations ordinaires et particulières reflète une recherche, une exploration ou une observation en fonction du reste du contexte.

5.4.6 *La volonté de modifier la situation*

Au delà de la spécifique mise en valeur, un type d'adaptation est la modification d'une composante de la situation comme la tâche ou la technique d'interaction. Ainsi si la tâche en cours est estimée globalement difficile, une modification de la technique d'interaction actuellement active est à envisager (Fig. 107a). Si un objectif est difficile à réaliser, sa modification doit également être envisagée plus fortement (Fig. 107b). Les suites données à cette volonté de modification dépendent du reste du contexte. Avec une modification désirée et des techniques différentes disponibles, un choix est demandé pour déterminer la meilleure dans la partie assistance. Ce choix peut être alors activé. Mais il pourrait également être présenté à l'utilisateur pour validation. De même proposer des paramétrages différents de la même technique est une modification moins brutale de l'interaction que l'on pourrait proposer.



(a) Difficulté générale de la tâche (b) Difficulté à effectuer un objectif

FIGURE 107: Volonté de modification de la technique en cours

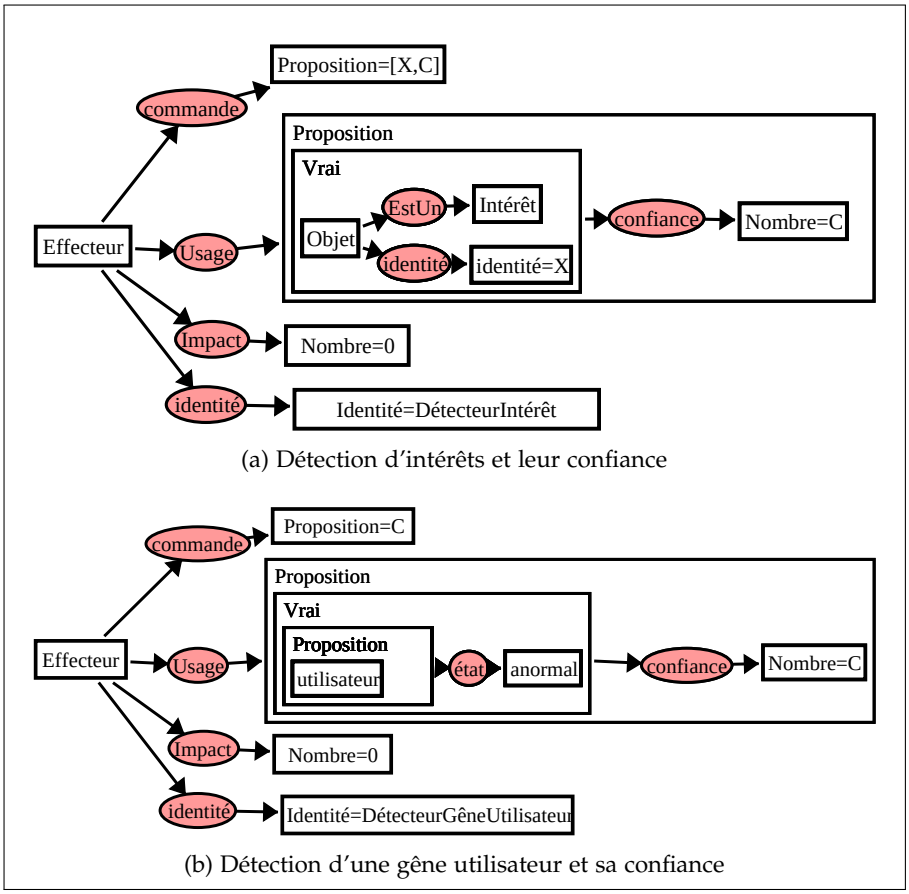
5.5 CRÉATION D'EFFECTEURS POUR L'ADAPTATION

Cette section aborde la dernière étape du processus de réaction : la réification de décisions décidées par le moteur. Ce sont donc des effecteurs contrôlés qui sont ici présentés. Mais, lorsque le moteur déclenche un effecteur, il ne connait pas précisément son effet. Ainsi certains effecteurs contrôlés ont un but différent de l'adaptation directe et permettent à l'application d'obtenir automatiquement des informations de la base de connaissance (section 5.5.1). Il faut ensuite créer une série d'effecteurs Virtools pour avoir des possibilités d'adaptations pratiques. Ces effecteurs sont ensuite classés selon leur usage probable : réactif (section 5.5.2) ou pro-actif (section 5.5.3). Des exemples de méta-effecteurs sont ensuite abordés (section 5.5.4).

5.5.1 *Effecteurs pour extérioriser le traitement sémantique*

Il est possible pour l'utilisateur, le concepteur ou l'application de questionner directement le moteur. Cela leur permet ainsi de bénéficier de ses capacités d'analyses. Obtenir des informations automatiquement peut se faire en créant un effecteur spécifique. C'est-à-dire qu'en fixant une requête comme étant l'*usage* d'un *effecteur*, le moteur le déclenchera potentiellement pour chaque possibilité d'unification, tout particulièrement si son impact est nul. Il est ainsi possible de rendre un comportement dans l'application dépendant d'une réponse du moteur sans que ce soit une adaptation affectant l'utilisateur.

Par exemple, il est intéressant pour analyser le comportement de l'utilisateur de pouvoir enregistrer automatiquement les différents intérêts ou difficultés estimés par le moteur au cours de son activité ainsi que la confiance en l'hypothèse. Il suffit pour l'application d'avoir accès à ces informations. Les descriptions sémantiques d'outils Fig. 108a pour les intérêts ou Fig. 108b sur l'apparition d'une gêne utilisateur le permettent. Sous l'application, elles ne déclencheront pas d'adaptations mais l'écriture d'un fichier. Ces effecteurs sont ainsi une version automatisée d'une requête spécifique à laquelle le moteur répond dès qu'une solution est disponible.



(a) Détection d'intérêts et leur confiance

(b) Détection d'une gêne utilisateur et sa confiance

FIGURE 108: Exemples de détecteur profitant de la fusion du moteur

5.5.2 *Effecteurs pour une adaptation réactive*

La mise en valeur d'intérêts est une forme d'assistance souvent réactive et cible des objets de l'environnement. Ces effecteurs sont utilisables quelle que soit la tâche en cours. En effet l'aide ajoutée n'influence par directement l'interaction et l'utilisateur peut les ignorer au besoin. Ainsi nos effecteurs Virtools permettent :

– la modification de la couleur d'un objet.

  L'effecteur permet de modifier le matériau d'un objet et de lui en associer un autre, de couleur et de transparence réglable. Cet outil nécessite la gestion d'une activation et d'une désactivation. A l'activation, les matériaux d'origine sont sauvegardés et remplacés. A la désactivation, les matériaux d'origine sont rétablis et la sauvegarde supprimée. L'outil, centralisé pour l'ensemble des objets, gère ainsi un tableau de collections de matériaux.

– le déclenchement de la pulsation d'un objet.

  L'effecteur modifie la taille de l'objet suivant une courbe non linéaire dépendant du temps. Cette valorisation n'agit pas uniquement sur la taille de l'objet mais ajoute également une composante de mouvement dépendant du temps, ce qui augmente d'autant plus la salience de l'objet.

– la création d'une courbe indiquant le chemin entre l'utilisateur et un objet.

  L'effecteur crée une trajectoire entre un objet source et un objet destination, en prenant en compte les obstacles de l'environnement. Ainsi celle-ci est créée peu à peu. Si un obstacle est rencontré sur la ligne droite entre le précédent point de la trajectoire et la destination, un nouveau point est ajouté le long de la normale à l'obstacle. La trajectoire peut ensuite être représentée par un tube, par des particules (ajoutant une composante de mouvement) ou un guide (un objet ou une attraction visuelle/haptique se déplaçant le long de la trajectoire). Chaque option peut être décrite plus ou moins précisément formant alors des outils différents.

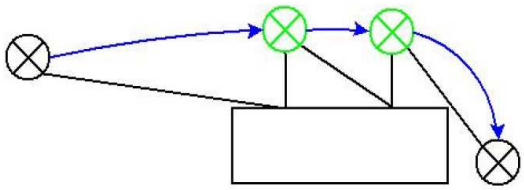


FIGURE 109: Principe de la création d'une trajectoire vers un objet

– Affichage d'une boussole indiquant la direction de l'objet.

  Une flèche 3D affichée sur l'interface utilisateur s'oriente en permanence vers un objet ciblé.

5.5.3 *Effecteurs pour une adaptation pro-active*

Les adaptations pro-actives sont des assistances permettant d'aider l'utilisateur à accomplir une tâche particulière. Il faut donc pouvoir détecter des situations d'usages adaptées avant la fin de la tâche. Ces adaptations peuvent également être déclenchées en tant que valorisation mais ont un effet sur des éléments essentiels de l'interaction : la caméra, la technique d'interaction, les représentants de l'utilisateur dans l'environnement etc. Ainsi nos effecteurs Virtools permettent :

– l'ajout d'une attraction ou d'une répulsion, visuelle (agissant sur le pointeur uniquement) ou haptique (agissant sur l'utilisateur avec le matériel adapté).

  L'outil permet d'appliquer une modification de position selon une vitesse dépendant de la distance vectorielle à un objet. Les options sont le sens de cette vitesse (attraction ou répulsion), sa norme (la force, le coefficient de proportionnalité appliqué à la distance), une distance réglant une zone de détection à hystérésis (permettant l'application ou non de la force, en évitant les effets de rebonds) ainsi que la commande externe appliquée à l'objet qui subira l'attraction. En effet l'objet attiré est souvent déjà commandé, comme le pointeur par un flystick par exemple. Ainsi le résultat de l'outil d'attraction sera la modification de la commande : une addition dans le cas d'une commande par vitesse, une addition pondérée dans le cas d'une commande par position de l'objet. C'est pourquoi pour être applicable, l'outil d'attraction nécessite une abstraction supplémentaire entre la commande directe et la commande

appliquée à l'objet (abstraction souvent déjà présente, par exemple pour régler le gain du matériel d'entrée). Enfin, l'outil sert également de capteur non contrôlé et indique lorsque l'attraction est effectivement ajoutée : lorsque l'objet sur lequel s'applique la force est suffisamment proche de l'objet l'exerçant.

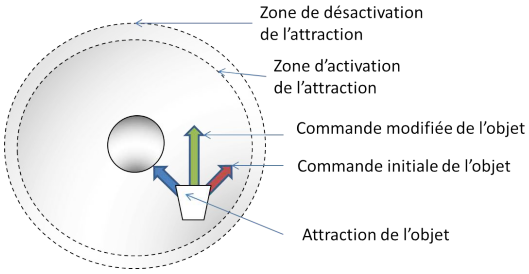


FIGURE 110: Principe de gestion de l'attraction d'un objet

– la modification du point de vue de la caméra.

  Cet effecteur modifie la direction et/ou la position de la caméra active. Il aligne la caméra sur un objet ou la déplace le long de cette direction. Ce dernier cas lors d'une vue à la première personne, effectue un zoom paramétré par un coefficient de grossissement. Les modifications se font progressivement avec une vitesse dépendant d'une fonction non linéaire afin de simuler un voyage physique avec une accélération de départ, une vitesse de croisière et une décélération à l'arrivée.

– le choix de la tâche active.

  Un effecteur qui permet de choisir la tâche élémentaire d'interaction active (sélection, manipulation, navigation). Bien sûr, la tâche est déterminée par l'intention de l'utilisateur. Néanmoins différentes interprétations de la commande utilisateur sont appliquées par Fly-over (section 5.3.2.2), afin de répondre au mieux aux besoins de chaque tâche. Désolidariser ces commandes de la détection automatique intégrée, ou plutôt ajouter un contrôle prioritaire permet de forcer la sélection de ces interprétations et donc finalement une sélection de la tâche active. Déclencher une manipulation nécessite d'expliciter un objet (ou un groupe d'objet) dans la commande.

– le choix de la technique d'interaction ;

  Un effecteur qui remplace la technique de sélection actuelle par celle de la main virtuelle, du ray-casting ou du 2D-picking.

5.5.4 *Exemples de méta-effecteurs*

Le moteur peut également avoir des réactions l'affectant directement en interne. Dans ce cas, les outils ne sont définis effectivement que sous PCG. En revanche une application peut définir de tels outils et les inscrire dans PCG pour obtenir ses

propres méta-adaptations. Deux méta-effecteurs on été créés permettant :

– le choix d'un niveau de réactions : difficile, normal, facile.

  Cela permet globalement de prendre plus facilement ou plus difficilement des décisions. Cela modifie à la fois le niveau de pertinence nécessaire pour chaque réaction ainsi que la quantité de réactions possibles. Ainsi le choix du niveau de réaction difficile (réciproquement normal ou facile) diminue (réciproquement réinitialise ou augmente) le niveau d'impact admissible total proportionnellement au niveau initialement défini et augmente (réciproquement réinitialise ou diminue) le niveau de ratio nécessaire pour garder une réaction.

– le choix du type de réactions : approchée ou non.

  Cela permet d'accélérer le processus de réaction en sélectionnant des réactions pertinentes mais qui ne sont pour autant les meilleures. Ce méta-adaptateur modifie les liens permettant d'obtenir la liste des réactions, avec un algorithme recherchant l'ensemble des possibilités avant de sélectionner les meilleures pour la version classique (Fig. 71), ou les sélectionne à mesure dès qu'une possibilité trouvée est pertinente, avec l'option de s'arrêter plus tôt, pour la version approchée (Fig. 72).

Les méta-adaptations n'ont pas encore été très poussées. Mais le moteur est très ouvert et permet de multiples modifications dynamiques. Notamment en modifiant des paramètres du moteurs, comme ci-dessus, parmi ceux présentés section 4.4.5.

5.5.5 *Discussion sur la gestion des outils*

La gestion des outils a permis d'illustrer des soucis inhérents à l'approche. La commande par message peut provoquer des problèmes lors de la perte ou de la répétition involontaire de ces messages. Par exemple, l'effecteur de changement de couleur ne recevait pas l'ensemble de ses commandes du à un problème sous Virtools. La perte d'un message de désactivation pouvait résulter en la réception de deux messages d'activations pour le même objet. Le matériau originel de l'objet était alors perdu et la désactivation suivante aboutissait à un objet sans matériau.

Cela illustre les erreurs pouvant résulter d'une approche parallèle. En effet, si deux activations ont pu être décidées, c'est du au fait que la connaissance de la scène par le moteur et sa réalité était différente. Ces erreurs peuvent survenir notamment à l'issue de la gestion interne de certains fonctionnements, comme de supposer l'état actif après activation. Mais cela permet également de grandement limiter les contraintes sur les outils ou l'application. Le problème a finalement été résolu par l'ajout d'un tampon supplémentaire pour les messages. Mais pour une application donnée, il reste possible d'ajouter une vérification ou une mise à niveau de la base de connaissance par les outils en cas de commande incohérente. Au pire, l'outil peut se supprimer lui-même de la base de connaissance.

La gestion des outils a donc soulevée de nombreux problème notamment pour définir ou obtenir leur commande. Cela a nécessité l'introduction de plusieurs concepts comme les états actif (section 4.5.2.1) ou disponible (section 4.5.2.2), le

choix (section 4.5.4.1) et notion d'agrégation (section 4.5.4.2). L'élaboration de commandes puis les règles d'assistances sont abordées dans la section suivante.

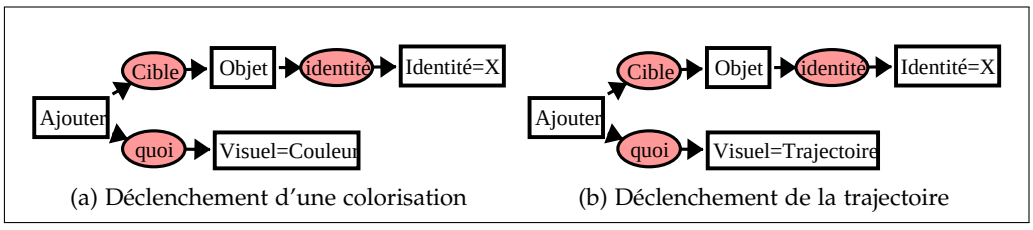
5.6 UTILISATION DU MOTEUR POUR L'ASSISTANCE

Une bonne description des outils est essentielle pour leur commande sémantique efficace afin d'obtenir une meilleure combinaison de règles. Son élaboration doit alors être soignée (section 5.6.1). Favoriser la combinaison passe aussi par la séparation d'éléments d'adaptations notamment selon les attributs des objets (section 5.6.2). Ces éléments ou des adaptions complètes peuvent avoir pour causes des situations plus complexes (section 5.6.3). Enfin des règles et cas du moteur affectent globalement l'ensemble des réactions (section 5.6.4).

5.6.1 *Définition sémantique des outils*

5.6.1.1 *Granularité des descriptions*

La définition sémantique des outils est effectuée par le concepteur. La gestion de cette liberté par le moteur rend l'expression de règles générales difficile. Le concepteur doit au besoin ajouter des règles pour relier l'usage de ses outils à l'expression de cas d'assistance déjà exprimés autrement. La taxonomie des assistances est donc très rapidement utile pour proposer au concepteur des points d'accroches. Cela permet également d'effectuer au mieux la description des outils proposés et de réduire la difficulté et le nombre de règles nécessaires pour les relier aux cas existants. Cette taxonomie permet notamment de définir la granularité nécessaire à la description des outils. En effet, la définition sémantique ne doit être ni trop précise ni trop générale. La bonne granularité doit inclure une description haut niveau de l'*usage* de l'outil. Il faut éviter les descriptions d'effets trop bas niveau, les détails d'implémentation comme la notion de création d'une flèche 3D associée à une navigation vers une cible (section 5.5.2). A plus haut niveau, cette flèche est la métaphore d'une boussole et permet d'indiquer à l'utilisateur une direction. Une description envisageable est ainsi la représentation de la direction entre deux entités, l'utilisateur et un objet ciblé. En revanche, décrire cet effet comme une aide à la navigation est trop haut niveau. Cela permet le déclenchement de l'outil mais ne permet pas une bonne discrimination des situations, où il serait le plus utile. La taxonomie progresse également lorsque l'on définit les règles utilisant ces outils.



(a) Déclenchement d'une colorisation (b) Déclenchement de la trajectoire

FIGURE 111: Exemples de commandes possibles mais peu adaptées

5.6.1.2 *Taxonomie de nos outils*

Les différents capteurs et effecteurs qui ont été créés doivent être exploités au mieux, notamment en évitant des règles ad-hoc pour chaque usage. Tenter d'en ef-

fectuer une classification permet de déterminer une meilleure description et ainsi un meilleur usage pratique. L'ensemble des outils, à l'exception du choix de technique de sélection (et des méta-adaptations, non classées), cible un objet (d'intérêt). Mais ont-ils tous réellement un impact sur l'objet même ? Quelles en sont les différentes caractéristiques ?

Le tableau Fig. 128 illustre une classification. Les caractéristiques discriminantes retenues sont : l'effet de l'action (sur l'objet ciblé, l'espace, l'interaction), l'effet dans le temps de l'action (limité ou illimité et avec ou sans variabilité), ainsi que la modalité utilisée (visuel, sonore ou haptique) et le le type d'action (e.g. modification ou représentation). La valeur de l'impact est déterminée par les catégories utilisées. Un effet est considéré variable si en l'absence d'entrée utilisateur, une différence peut être perçue par celui-ci (indépendamment de sa durée et jouant sur sa salience). Ces catégories ne sont pas exhaustives (l'effet peut aussi impacter ou non l'interface utilisateur etc.).

Principe	Entité impactée par l'effet			Type d'effet dans le temps			Modalité utilisée par l'effet			Description de l'action	Valeur de l'impact
Reification	Objet	Espace	Interaction	Illimité	Avec variabilité	limité	Visuel	Sonore	Haptique	Sous-concepts utilisés	Selon les catégories
Couleur	X			X			X			Représentation (Identité)	0,2
Pulsation	X			X	X		X			Représentation (Identité)	0,3
Boussole		X		X	X		X			Représentation (Direction)	0,2
Bip		X		X	X			X		Représentation (Distance)	0,3
Trajectoire		X		X	X		X			Représentation (Trajectoire)	0,2-0,3
Attraction		X	X	X	X		X		X	Modification (Distance)	0,9-1
POV		X	X		X	X	X			Modification (Direction)	0,7
Zoom		X	X		X	X	X			Modification (Distance)	0,7
Choix tâche			X	X						Modification (Tâche)	0,8
Choix technique			X	X						Modification (Technique)	0,8

X Options selon l'implémentation de l'outil

FIGURE 112: Classification des effecteurs

VARIABILITÉ DU CLASSEMENT DES OUTILS Certains outils peuvent être multi-classés. Leurs caractéristiques dépendent ainsi de l'implémentation choisie et/ou les outils peuvent proposer des choix selon leur commande. Ainsi l'attraction peut être uniquement visuelle (déplacement du pointeur) ou haptique (une force est perceptible via un outil spécifique pour l'utilisateur) et dans les deux cas, affecte simultanément l'espace et l'interaction. Une attraction ne portant pas sur un élément contrôlé directement par l'utilisateur ne porterait alors plus que sur l'espace. De même, l'outil créant la trajectoire peut le faire via des particules ou un guide visuel, introduisant une variabilité de l'effet, ou alors par un affichage fixe. Enfin, l'outil de colorisation est décrit comme touchant à la représentation de l'identité de l'objet. Néanmoins il est également possible de le décrire comme modifiant un attribut de l'objet (et la pulsation comme modifiant la taille de l'objet) et de relier cette possibilité à la représentation de l'objet par une règle supplémentaire. La taxonomie

se reflète alors dans les règles plutôt que dans la définition de l'usage de l'outil.

ENTITÉ IMPACTÉE PAR L'OUTILS Ainsi si tous les choix, et souvent les paramétrages de l'adaptation, dépendent de l'objet d'intérêt même, la plupart ne le modifient pas. La plupart des outils modifient ainsi l'espace (par représentation ou modification). Par exemple l'outil de trajectoire créé une représentation visuelle de l'espace reliant l'utilisateur et l'objet d'intérêt. De même pour l'attraction et le bip sonore, l'objet lui-même n'est pas modifié, c'est bel et bien l'espace autour de l'objet qui acquiert une propriété. Le fait qu'elle soit parfois variable dans le temps, seuillée (attraction sous une certaine distance) ou non visuelle (modalité sonore) empêche peut-être cette compréhension de prime abord. Il faut interagir avec l'espace pour en percevoir la propriété, ce qui est la définition d'un champ physique. Ainsi au final, ces effecteurs créent donc un champ d'attraction et un champ sonore. Le bip et l'attraction sont en fait des réifications très proches mais avec une modalité différente. Néanmoins l'attraction affecte souvent l'interaction et prend en compte la distance vectorielle. La modalité sonore dans notre implémentation ne prend en compte que la norme de la distance (le son pourrait être également directionnel).

CRÉATION GUIDÉE DE NOUVEAUX OUTILS Il est d'ailleurs intéressant de tenter d'obtenir d'autres outils en modifiant des catégories associées. Ainsi rendre la boussole haptique ne la transforme pas en attraction, car la force de la boussole haptique ne dépend pas de la distance vectorielle mais uniquement de la direction. Elle peut être implémentée à l'aide d'une force constante dans la direction de l'objet ou par une force proportionnelle à l'angle entre deux directions. C'est donc bien un outil différent de l'attraction précédente. De même une trajectoire associée à la modalité haptique ajoute une attraction, non pas dirigée vers l'objet mais vers le point de la courbe le plus proche. En dessous d'une certaine distance, le point considéré se déplace alors le long de la courbe permettant de la suivre. Ainsi si ces outils modifient tous une propriété de l'espace, la caractéristique de l'espace modifiée (direction, trajectoire, distance ou distance vectorielle) change réellement l'effet pratique. Ajouter une composante variable dans le temps au changement de couleur est également possible, et fait alors scintiller l'objet. Rendre dépendant la variabilité dans le temps de la pulsation (sa fréquence) de la distance combine la modification de l'identité et la représentation de la distance, avec une modalité visuelle. Représenter l'identité de l'objet peut également se faire autrement que par une modalité visuelle : faire ressentir le relief de l'objet de manière haptique ou le dénommer de manière sonore.

DÉTERMINATION GUIDÉE DE L'IMPACT D'UN OUTIL Cette classification permet aussi de définir plus facilement l'impact initial. On peut considérer un impact élémentaire par catégorie puis les additionner pour ainsi obtenir les valeurs du tableau (Fig. 128). L'impact maximal est limité à 1. Ainsi un apport par composante de 0.1 est faible et 0.3 élevé. Le résultat est toujours un ordre de grandeur mais introduit un cohérence dans leur estimation :

- 0.2 pour une représentation (moyen) ;
- 0.3 pour une modification (élevé) ;
- 0.2 pour une modification illimitée (moyen mais cumulatif) ;
- 0.1 pour une composante variable dans le temps (faible) ;

– 0.3 pour un impact sur l'interaction (élevé) ;
– 0.3 pour une modalité haptique (élevé) ;

5.6.1.3 *Descriptions des usages des outils*

Les descriptions d'usages sont ainsi obtenues directement de la classification illustrée Fig. 113. L'action, l'entité qui la subie, l'objet ciblé et la modalité utilisée sont des caractéristiques principales et associées directement à des branches de CGs. Le caractère illimité ou limité est directement explicité par l'activation possible de l'outil. Une relation qualifiant l'effet dans le temps peut être introduite. Bien que l'activation porte déjà l'idée d'une limite ou non de l'effet suite à la commande, c'est une piste d'évolution pour conserver l'ensemble des caractéristiques retenues sur un même niveau [10] de description. En effet, la réflexion au sein du moteur se fait sur les descriptions contenues dans l'activation qui n'est décidée qu'au final. Cela permettra également de mieux intégrer la notion de variabilité dans le temps.

Des parties de ces descriptions sont en soi optionnelles : celles emboitées dans les concepts d'action et d'entité modifiée. Ces informations permettent au moteur des interprétations supplémentaires sans pour autant s'imposer dans la réflexion. En effet la demande d'une modification de distance ciblant un objet sans plus de précision suffit à activer l'outil de zoom (Fig. 113a). L'avantage de ces descriptions est ainsi de permettre la réflexion et l'influence d'ensemble sans aucunes règles supplémentaires. Ainsi il est possible de discuter pour une cible :
– de la représentation de l'espace : affectent l'outil de trajectoire (Fig. 113b) et de bip sonore (Fig. 113c).
– de modification de l'espace : affectent l'outil de zoom (Fig. 113a) et d'attraction (Fig. 113c)
– d'action sur l'espace : affectent l'ensemble des outils représentés (Fig. 113).

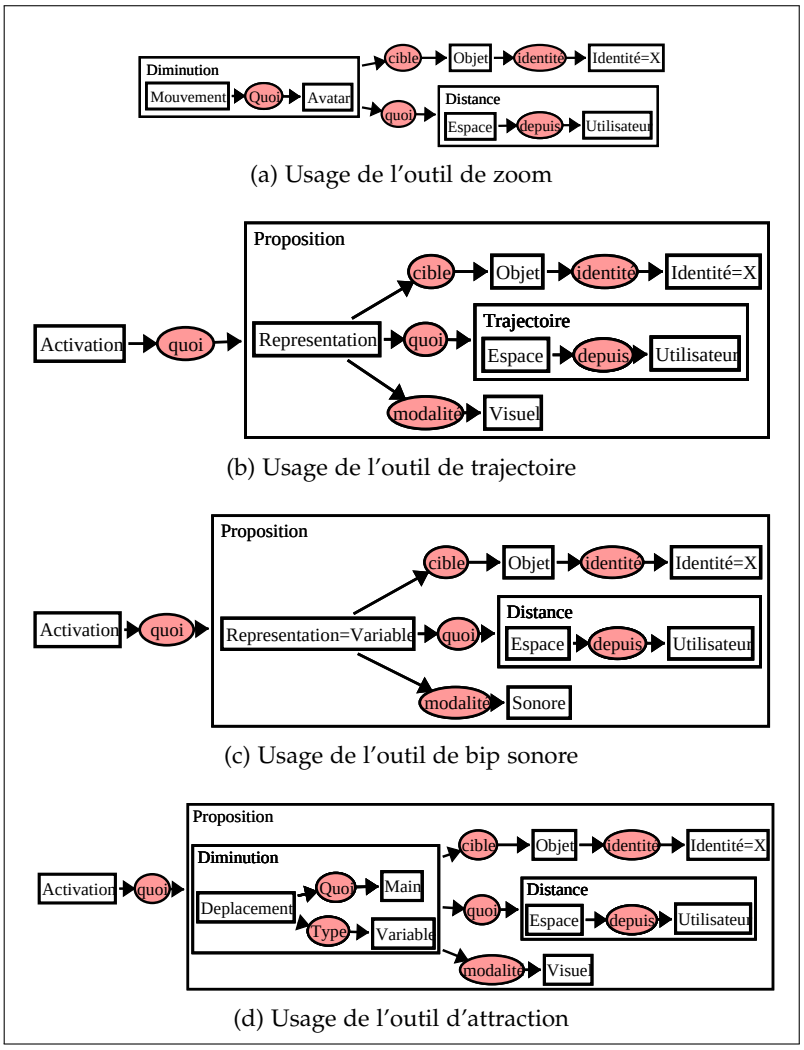
Ainsi, la variabilité dans le temps est ici optionnelle et ne pas la préciser n'est pas bloquant. Elle est déjà en partie reflétée par l'impact de l'outil. C'est une information qu'il peut être néanmoins utile de connaitre pour décider des adaptations. Un objet à la visibilité très basse peut alors être valorisé de manière préférentielle avec une représentation variable. Des confiances différentes peuvent ainsi favoriser l'outil avec variabilité malgré son impact plus important.

De même l'effet simultané sur l'interaction de certaines de nos adaptations est ici décrit comme secondaire. Cette simultanéité est déjà, comme pour la variabilité, présente dans la valeur de l'impact. Il reste possible lors de la réflexion d'obtenir à l'aide des descriptions emboitées que ces outils ont un impact sur l'interaction (via les règle d'*impact* entre *propositions* section 4.5.3.1). Cela reste ainsi facile de valoriser ou dévaloriser une action selon ces descriptions optionnelles.

Si l'on veut rendre la simultanéité obligatoire sans rajouter [11] une deuxième relation *quoi*, il faut avoir une liste de deux descriptions (ou plus) dans l'usage de l'outil chacune traitant d'une entité. L'ensemble doit être vrai pour déclencher l'outil. Par exemple, pour le zoom ou l'attraction visuel, il faut que la base de connaissance

10. Il faut alors ajouter de manière générale l'équivalence entre une activation et le déclenchement d'un effet illimité.
11. Voir discussion sur les relations simultanées dans le cas des attributs section 4.3.2.4.

puisse décider simultanément d'une modification de l'interaction et de l'espace pour les utiliser.



(a) Usage de l'outil de zoom

(b) Usage de l'outil de trajectoire

(c) Usage de l'outil de bip sonore

(d) Usage de l'outil d'attraction

FIGURE 113: Exemples de descriptions d'usages d'outils

Ainsi le choix de la description d'un outil est subtil. Mais une bonne description ouvre de bien meilleures possibilités initiales et combinaisons pour le raisonnement. Il est cependant difficile de ne pas trop spécialiser nos descriptions via l'ajout de trop nombreuses relations tout en donnant au moteur la possibilité d'exploiter des informations supplémentaires disponibles. Car toute relation supplémentaire est éventuellement bloquante si le moteur ne sais pas l'utiliser. De plus il est souvent intéressant de ne réfléchir que sur une partie de l'adaptation. Par exemple, considérer l'apport d'une modalité particulière. Mais le concepteur doit alors prévoir la structure de ses outils et ainsi préciser l'ensemble des branches qui peuvent apparaitre dans toutes combinaisons. Ces problèmes ont menés à l'introduction de la notion d'agrégation (section 4.5.4.2). Elle permet de lever ces limitations et de continuer à procéder graduellement et simplement, comme nous le verrons ci-dessous. Par la suite, les règles déclenchant les assistances via ces outils sont présentées (avec ou sans cette notion d'agrégation).

5.6.2 *Définition et combinaison d'éléments d'adaptation dépendant des attributs*

5.6.2.1 *Éléments d'adaptations par attribut*

Selon la situation, les adaptations pertinentes à mettre en oeuvre sont différentes. En considérant uniquement les attributs d'un objet, il est possible de considérer des *éléments* d'*adaptation* le concernant (Fig. 114). Ainsi si un objet est visible, une adaptation peut utiliser une modalité visuelle. Si celui-ci est caché il est préférable d'utiliser une modalité sonore. Pour éviter des risques de collisions, la modalité haptique n'est envisagée que dans le cas où l'entourage est faible. Si l'objet est difficile à différencier ou à repérer (avec une faible visibilité angulaire, entouré d'autres objets etc.), modifier sa représentation est envisagé. Mais la représentation de l'espace entre celui-ci et l'utilisateur est également un bon moyen d'aider à le repérer (objet éloigné, caché etc.). Des règles peuvent ainsi définir la confiance de ces morceaux d'adaptations séparément. En fonction de la situation, leur combinaison permet de déclencher l'outil le plus adapté.

Attribut	Modification	Représentation	Modalité
Caché	Modification de la position /direction de l'utilisateur	Représentation de l'espace depuis l'utilisateur	Sonore
Visible		Représentation de l'identité	Visuelle
Distance faible			Haptique
Distance élevée	Diminution de la distance depuis l'utilisateur	Représentation de l'espace depuis l'utilisateur	
Entourage faible			Haptique
Entourage élevée		Représentation de l'identité	Visuelle
Visibilité angulaire faible		Représentation de l'identité	
Visibilité angulaire élevée			Visuelle

FIGURE 114: Tableau récapitulatif liant un attribut et les éléments d'adaptations

Par exemple supposons l'ensemble des capteurs d'attributs présents et un objet proche, de petite taille, plutôt caché derrière de nombreux obstacles. Chaque attribut mesuré augmente la confiance en certains éléments d'adaptations. Ainsi chaque élément est favorisé ou défavorisé par les différentes règles :

- 2 en faveur d'une modalité sonore (objet caché et à visibilité angulaire faible) ;
- 1 en faveur et 2 en défaveur d'un modalité visuelle (objet à entourage élevée mais avec une faible visibilité angulaire et peu visible) ;
- 1 en faveur et 1 en défaveur d'une modalité haptique (objet à distance faible mais à entourage élevée) ;
- 2 en faveur et 1 en défaveur d'une représentation de l'identité (objet à entourage élevé et faible visibilité angulaire mais partiellement caché) ;
- 1 en faveur et 1 en défaveur de la représentation de l'espace (objet caché mais proche) ;
- 1 en faveur de la modification de la position ou de la direction de l'utilisateur ;

Il est difficile en soi de prédire l'assistance choisie dans cette situation car cela dépend de la confiance dans les règles et dans les attributs, du fait que la fusion soit optimiste et augmente donc avec le nombre de règles répondant, mais aussi des

différents outils disponibles ainsi que de leur impact différent. Néanmoins, toute chose égale par ailleurs, la modalité sonore ainsi que la représentation de l'identité sont favorisées. Une description de l'objet pourrait ainsi être lue si un tel outil était disponible. Avec les effecteurs créés, le" bip" traitant de la représentation de l'espace avec une modalité sonore est le plus adapté. Si l'impact total est suffisant, il est également possible qu'une représentation visuelle soit ajoutée (une pulsation ou une colorisation moins coûteuses). L'assistance est bien intéressante : l'utilisateur peut ainsi repérer globalement la localisation de l'objet grâce au son et le détecte ensuite plus rapidement malgré l'entourage grâce à l'aide visuelle.

5.6.2.2 *Règles pour les éléments d'adaptations*

Ces *éléments* sont donc définis dans des *règles* liant *attribut* et structure d'*adaptation*. Deux possibilités sont illustrés. La première tient compte de la structure des outils Fig 115 tandis que la deuxième n'exprime que l'essentiel Fig 116. La première permet d'obtenir directement toutes les combinaisons désirables sous certaines conditions quant à la structure des outils et du contexte. La deuxième permet d'exprimer uniquement ce qui est important et pose moins de risque d'erreur mais nécessite la notion d'agrégation pour être exploitable.

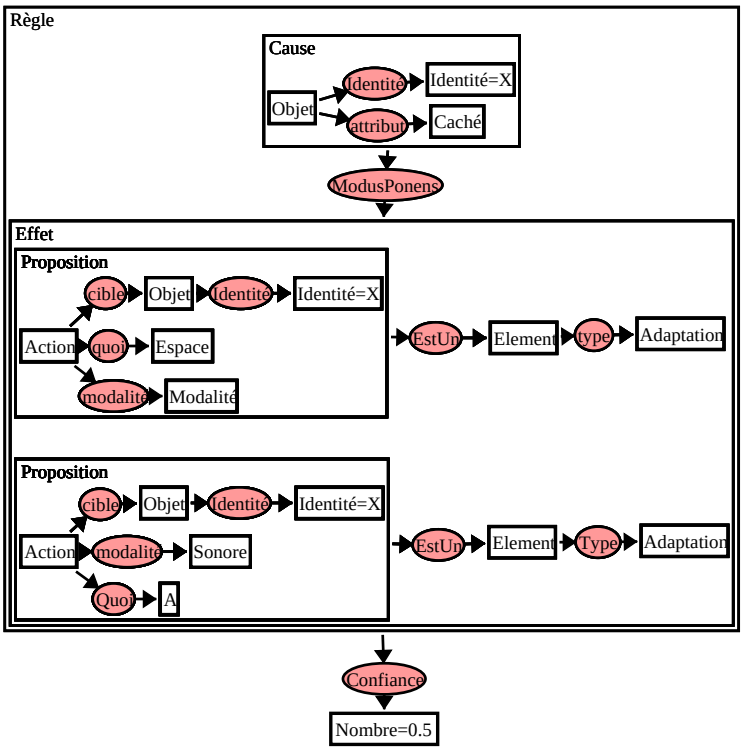


FIGURE 115: Définition d'un élément d'adaptation avec conditions sur le contexte

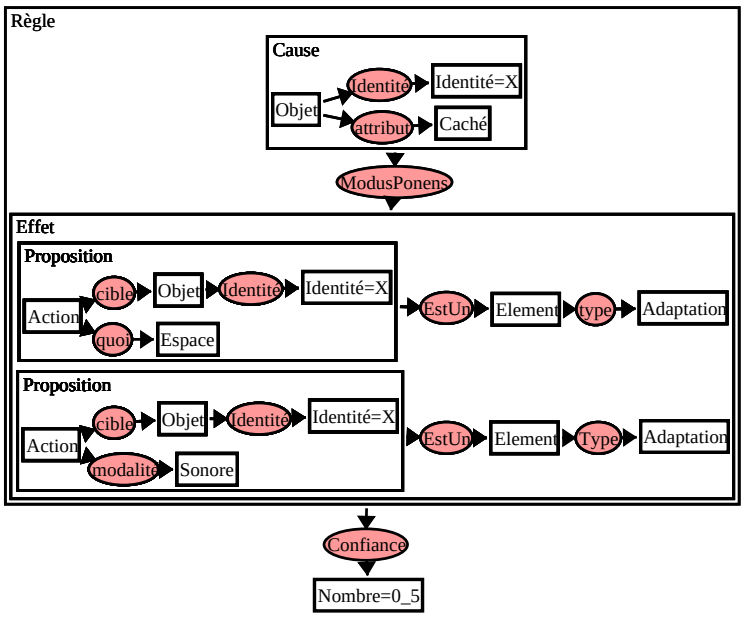


FIGURE 116: Définition d'un élément d'adaptation isolé mais d'usage général avec l'agrégation

Les règles permettant l'utilisation des éléments exprimés sont donc également différentes. La règle Fig 117 est suffisante pour les éléments d'adaptations dans un contexte maitrisé tandis que la règle Fig 118 est nécessaire pour utiliser la deuxième expression des éléments. Cette dernière est en revanche également compatible avec la première expression de ces éléments et plus généralement permet de tenter de récréer la structure des outils à partir des éléments d'adaptation. Les deux règles ont une pré-condition vérifiant que l'adaptation recherchée cible un objet et ajoute comme cause que l'on veuille le valoriser. Cette formulation permet de garder les éléments d'adaptations indépendant de l'usage qui en est fait.

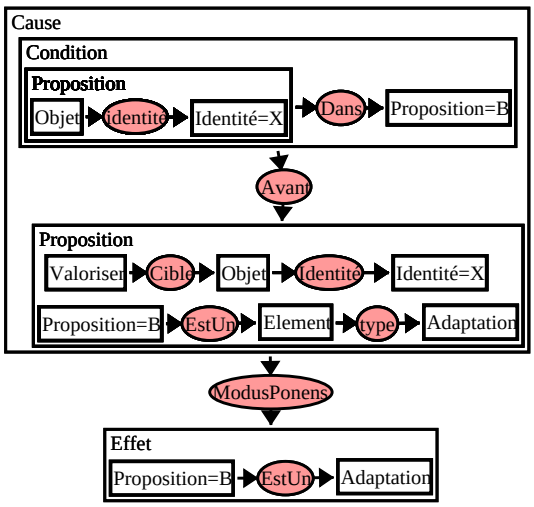


FIGURE 117: Règle fonctionnelle pour les éléments d'adaptation avec conditions sur le contexte

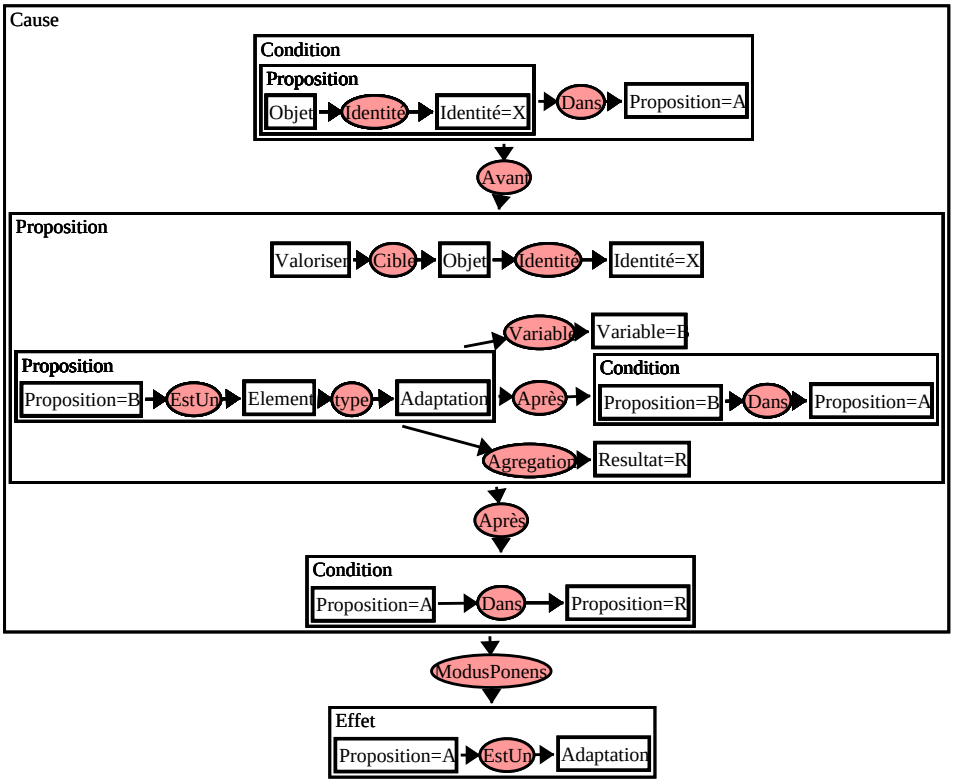


FIGURE 118: Règle générique pour la combinaison d'éléments d'adaptation

Le fonctionnement de la règle Fig. 118 est illustrée Fig. 119. La structure d'agrégation (section 4.5.4.2) répertorie les éléments d'adaptation et ne conserve que ceux compatibles avec l'adaptation, c'est-à-dire capables de répondre au moins partiellement à la demande. Ils sont alors agrégés et le résultat est vérifié afin de s'assurer avoir pu reconstruire au moins l'adaptation dans sa totalité (si ce n'est plus). La confiance de la structure d'agrégation est obtenue comme celle d'une règle unitaire qui aurait pour causes les éléments utilisés.

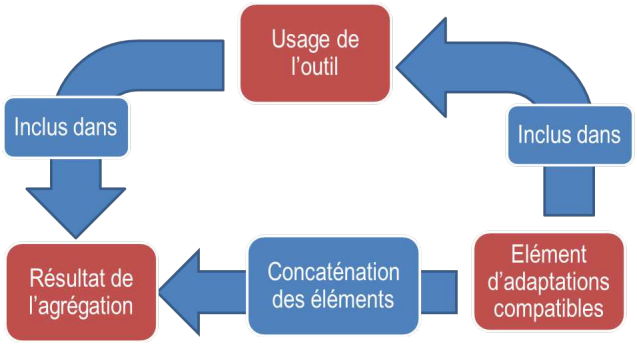


FIGURE 119: Illustration du fonctionnement de la règle Fig118

5.6.2.3 *Importance de la notion d'agrégation*

L'usage de l'agrégation semble plus complexe et est également plus lent d'exécution. Mais le raisonnement implémenté est fondamentalement différent et est général. Les possibilités de combinaisons offertes par la structure sans agrégation présentée Fig. 115 sont virtuellement les mêmes qu'avec l'agrégation sous trois conditions :

– la structure utilisée par les outils possède autant ou moins de relations que celles présentes dans les éléments d'adaptations ;
– l'ensemble des informations permettant de statuer sur les éléments d'adaptations sont présents dans la base.
– il existe un élément d'adaptation pertinent pour chaque partie de la structure.

Ces deux dernières conditions sont importantes et lourdes. En effet, par convenance d'écriture nous utilisons des branches générales pour permettre les combinaisons désirées. Mais nous avons alors décrit plus d'informations que nécessaire et souvent plus d'informations que prévu. Supposons par exemple qu'au vu du contexte, un seul élément d'adaptation soit dans la base précisant la modalité visuelle avec une stucture complétée par des relations générales pour satisfaire l'outil (à l'image de la modalité sonore Fig. 115). Quelle que soient l'action ou l'entité affectées, pourvu que la modalité visuelle soit présente, la réaction devient possible uniquement avec cet élément. En effet nous n'avons pas statuer réellement sur la pertinence de l'élément seul et cela peut provoquer de graves fautes de raisonnement si ce n'est pas maitrisé. Plusieurs moyens sont alors possible pour se passer néanmoins de l'agrégation.

– utiliser des structures générales (comme présentées Fig. 115) mais s'assurer que l'ensemble des outils que l'on puisse déclencher soient valides avec les informations disponibles. Dans notre cas l'ensemble des valorisations sont envisageables dès lors que l'on peut obtenir l'ensemble des attributs simultanément (ce que fait notre capteur par défaut). Ainsi chaque partie de la structure possible est décrite par au moins un élément et la combinaison finale nous permet d'obtenir la confiance adaptée pour chaque outil.
– utiliser des structures complètes mais spécifiques. Il faut préciser l'ensemble des usages où l'on souhaite favoriser la modalité visuelle lorsque les seules informations disponibles sont celles des causes de l'élément d'adaptation considéré.

Ainsi sans agrégation, les éléments d'adaptation sont soit rapides à écrire et à exécuter mais erronés en toute généralité, soit rapide à exécuter mais nécessitent au concepteur d'être exhaustif vis-à-vis de chaque caractéristique. Dans les deux cas, l'ensemble est difficile à étendre d'autant plus qu'il est dépendant d'une structure maximale des outils commandés. Au contraire l'agrégation offre de nombreux avantages malgré son exécution plus lente et permet réellement la réflexion sur les éléments séparés :

– Elle est générale et permet de combiner n'importe quel élément pertinent. Chaque branche possible d'une structure d'outils peut être définie seule ou avec d'autres dans un élément d'adaptation selon les besoins. Il est inutile de préciser des éléments non pertinents.
– Elle s'assure d'obtenir des combinaisons valides. Si une information manque pour construire l'usage d'un outil, celui-ci ne peut pas être déclenché. Les combinaisons sont naturelles et fonctionnelles sans risque d'erreur.
– Elle permet d'étendre le raisonnement sans modification supplémentaire. Étendre la structure des outils avec ajout d'une relation ne rend pas caduque les anciens éléments. Chaque élément nouveau n'a qu'a être précisé, indépendamment des autres ou non.

Il est possible d'ajouter des éléments génériques à très faible confiance pour compléter des informations manquantes. Par exemple, décrire qu'un élément d'adap-

tation (qui statue sur une action générique, avec une modalité générique, etc.,) utilisant un concept parent soit très faiblement vrai. Dès lors, si l'ensemble des capteurs permettant la déduction des modalités pertinentes sont absents, l'agrégation peut néanmoins recréer n'importe quelle structure. Cependant pour qu'une construction aie une confiance correcte, il faut que les éléments obtenus autrement que par cette structure générique aient une très bonne confiance. Cela permet au moteur de s'adapter ainsi facilement non seulement aux différentes effecteurs mais également aux différents capteurs disponibles.

Il est envisageable de tenter la construction d'un processus utilisant l'agrégation pour générer, à partir des éléments d'adaptations décrits, l'ensemble des causes, avec une structure complète et correcte, compatibles avec les outils disponibles. Ce serait alors une étape préalable pour chaque application connaissant sa liste d'outils et qui permet d'obtenir de manière générale la version des règles rapides, et toujours sémantiques, à partir de notre description de l'essentiel.

5.6.3 *Règles de gestion de situation complexe*

Précédemment, les éléments d'adaptations ont été déterminés à partir d'un seul attribut de l'objet visé, et ont permis déjà de prévoir des valorisations. Mais de nombreux éléments d'adaptations peuvent être prévus en combinant diverses informations de contexte (résumées Fig. 120). Cela peut être la combinaison d'attributs : si un objet peu couvert par des obstacles n'est pourtant pas visible, il peut être dans le dos de l'utilisateur amenant à une assistance modifiant sa direction. S'il est de plus éloigné, il est possible qu'il ne soit pas affichable car trop loin, il faut alors modifier la position de l'utilisateur. Autrement en connaissant un objectif donné lié à un objet (e.g. la sélection d'un objet) et sachant l'objectif difficile (dû notamment à la technique d'interaction et aux circonstances, section 5.4.1), il est possible de prévoir plusieurs assistances en fonction des attributs de l'objet. Ainsi si l'objet est proche et que l'objectif est sa sélection on peut tenter d'attirer l'utilisateur (usage d'haptique, diminution de la distance) voire même d'effectuer la sélection en activant la manipulation de l'objet. Si la distance est élevée ou si la visibilité angulaire est faible, la manipulation directe n'est pas envisagée car il est probable que le problème persiste autrement (c'est alors la manipulation qui devient difficile). Mais il est envisageable d'effectuer alors une part de la navigation vers l'objet pour aider à la future sélection.

Causes	Modification	Représentation	Modalité
Objet caché + couverture faible + distance élevée	Modifier la position de l'utilisateur		
Objet caché + couverture faible	Modifier la direction de l'utilisateur		
Sélection objet difficile+ distance faible	Diminution de la distance depuis l'utilisateur		Haptique
	Activer la manipulation de l'objet		
Sélection objet difficile+ distance élevée	Diminution de la distance depuis l'utilisateur		
Sélection objet difficile + visibilité angulaire faible	Diminution de la distance depuis l'utilisateur		
Navigation vers un objet difficile		Représentation de l'espace depuis l'utilisateur	

FIGURE 120: Tableau récapitulatif liant une situation et des éléments d'adaptations

Enfin d'autres adaptations ne sont pas exprimées comme élément mais directement comme des adaptations complètes. Les lien de causalité sont résumés Fig. 121 et les règles représentées Fig 122.

Causes	Modification
Tâche difficile + technique différente à utiliser	Choix de la technique d'interaction
Tâche est facile + tâche est un objectif	Activer la tâche
Tâche liée à un objet est facile + Tâche liée à un objet est un objectif	Effectuer la tâche

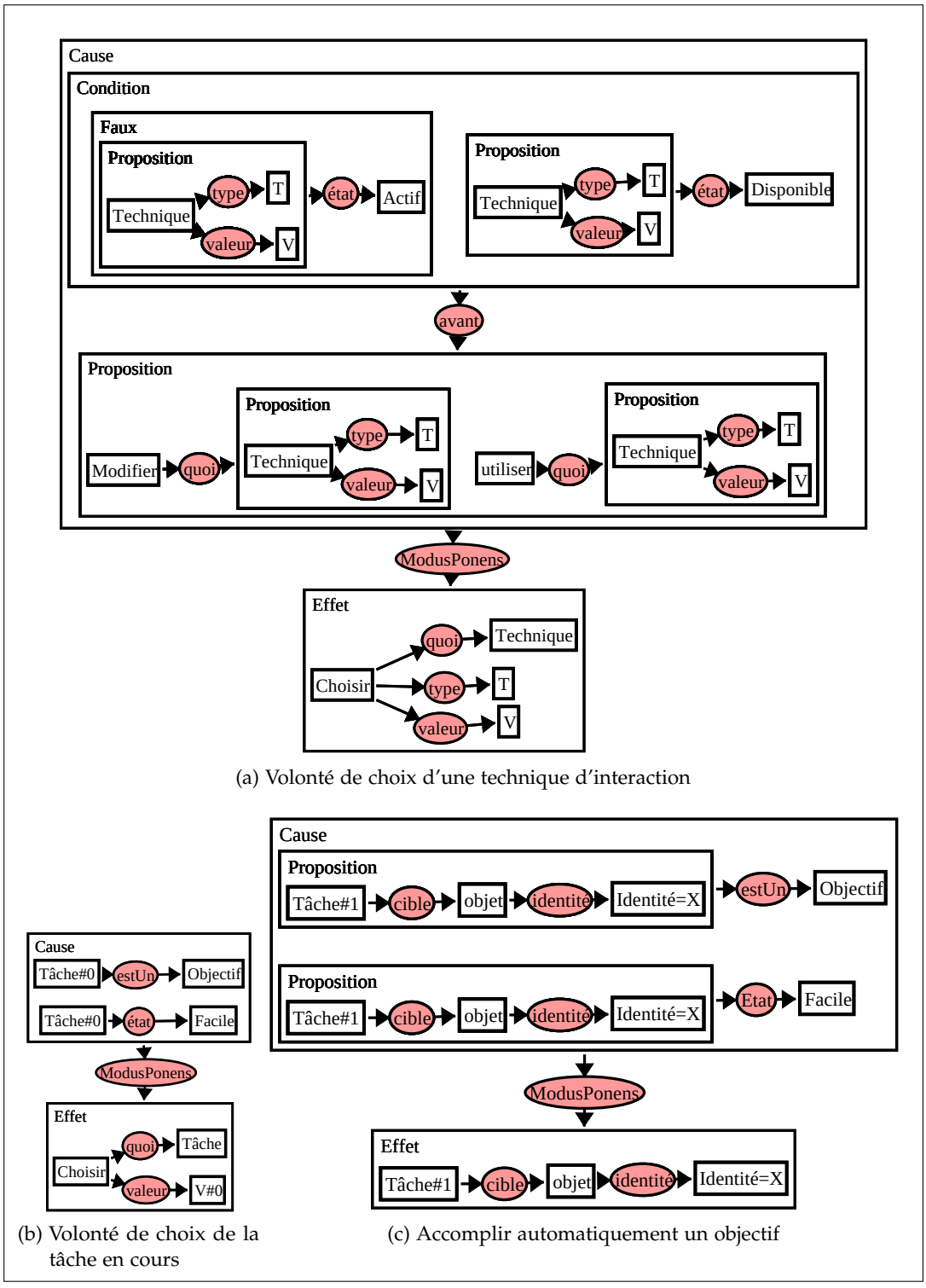
FIGURE 121: Tableau récapitulatif liant une situation et une adaptation

On peut effectuer le choix d'une technique (Fig. 122a). La volonté de choisir une technique (affectant le choix final section 4.5.4.1) dépend de la volonté de son usage dans la situation actuelle et de la volonté de modification de la technique courante (selon la difficulté de la tâche section 5.4.3). Pour la volonté de choisir, et donc de conserver, la technique active, une autre règle a plutôt comme cause le fait que l'on ne veuille pas modifier la technique courante. Dans tous les cas, pour être considérée une technique doit au préalable être disponible.

De manière similaire, si une tâche est globalement facile tout en étant un objectif, il est envisagé de choisir cette tâche (Fig. 122b). Le test de facilité est ajouté pour éviter que la tâche ne soit en permanence imposée par l'objectif. Par exemple, si l'objectif est la sélection (sans plus de précision, potentiellement de nombreux objets) et que la technique actuelle permet de gérer facilement les objets d'intérêts, ou plus généralement si la tâche est considérée facile au vu des circonstances, il est envisageable de l'entamer. C'est-à-dire de sélectionner le comportement associé à cette tâche. En revanche ne plus pouvoir se déplacer alors que les objets d'intérêt sont lointains et que la technique d'interaction active est la main virtuelle serait dérangeant.

Enfin, si un objectif détaillé est connu et qu'il est considéré comme facile, une adaptation possible est de le réaliser (ou en partie) à la place de l'utilisateur (Fig. 122c). De même, cela est limité aux tâches faciles pour plusieurs raisons. Premièrement la réalisation automatique de la tâche n'est pas toujours possible dans des cas difficiles, selon les implémentations de l'effecteur. De plus, accomplir un objectif à la place de l'utilisateur peut être perturbant. Par exemple, pour la navigation automatique dans un cas difficile (objet très éloigné, mouvement complexe), les changements sont grands et le risque de perturber l'utilisateur tout autant. De même, la sélection automatique dans un cas difficile (objet derrière un obstacle, ou très éloigné) risque de déclencher la manipulation d'un objet peu accessible, voire qui n'a pas encore repéré par l'utilisateur. C'est pourquoi on limite l'accomplissement automatique d'un objectif aux cas faciles : afin d'accélérer l'interaction sans perdre l'utilisateur. Et il est également beaucoup plus aisé de réaliser des implémentations d'effecteurs compatibles. Ainsi sélectionner automatiquement un objet proche ou déclencher un dernier virage automatiquement pour la navigation sont envisageables. Il serait intéressant plus généralement d'être capable de décomposer l'accomplissement d'un objectif en sous-étape pour assister l'utilisateur également

dans des cas complexes (cela touche alors notamment au planning).



(a) Volonté de choix d'une technique d'interaction

(b) Volonté de choix de la tâche en cours

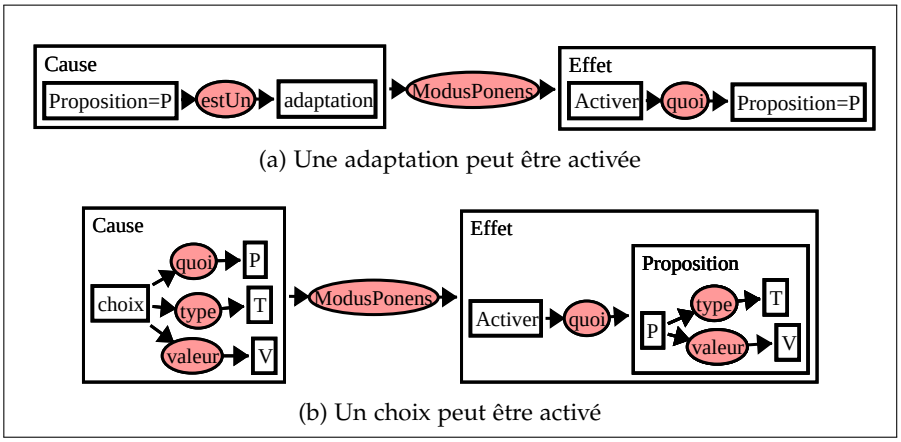
(c) Accomplir automatiquement un objectif

FIGURE 122: Gestion du choix et de l'activation d'une technique d'interaction

5.6.4 *Gestion d'influences globales du raisonnement*

5.6.4.1 *Gestion des activations et des désactivations*

L'activation est une notion qui permet au moteur de gérer l'impact et de reconnaitre facilement une décision durable. L'activation peut être imposée dans une situation. Autrement c'est pour nous une option envisageable sans condition supplémentaire, à partir d'adaptations explicites (pouvant être issues d'agrégation d'éléments, Fig.123a) ou de choix (Fig.123b) Ainsi, la plupart des adaptations considérées peuvent être activées par défaut. Mais il est envisageable d'introduire des confiances différentes en fonction du traitement final. Ainsi il est possible de présenter un choix à l'utilisateur, plutôt que de l'activer. Cette option permet à l'utilisateur de valider le choix qui semble cohérent, et aura alors une confiance plus grande que l'activation automatique. De même, un effecteur peut définir la présentation de l'ensemble des volontés de choix dépassant un certain seuil de confiance plutôt que de ne présenter que le meilleur selon le moteur. Ainsi le raisonnement interne du moteur est accessible par les effecteurs en ne rajoutant que la règle finale d'implémentation.



(a) Une adaptation peut être activée

(b) Un choix peut être activé

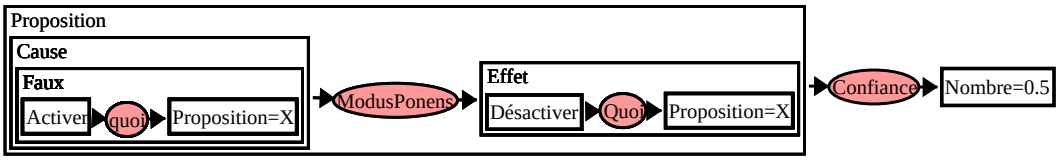
FIGURE 123: Gestion d'activations

Les adaptations activées doivent parfois être désactivées et plusieurs règles génériques sont prévues. Par exemple, si dans le contexte actuel, une réaction ne serait pas activée, sa désactivation est envisagée (Fig.124a). Bien que vraie par défaut (usage du faux), déclencher une désactivation via la volonté de désactiver vérifie au préalable l'état actif (section 4.5.2.1). La faible confiance de la règle permet ne pas enlever trop facilement une aide active qui serait un peu moins adéquate actuellement, afin d'éviter le changement constant des assistances pour de faibles variations de contexte. Notamment car l'impact d'une désactivation est souvent plus faible que celui d'une activation. Une autre règle de confiance unitaire (Fig. 124b), permet la désactivation d'une adaptation active jugée anormale (par exemple un mouvement anormal en présence d'une modalité haptique, section 5.4.1).

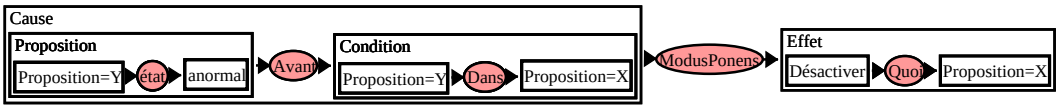
Des règles peuvent préciser des désactivations spécifiques. Par exemple, une règle précise la volonté de désactiver une adaptation ciblant un objet que l'on ne souhaite pas valoriser (Fig. 124c). La volonté de valoriser peut n'être plus vraie lorsque l'objet n'est plus un intérêt, lorsque la valorisation est implicite ou terminée (section 5.4.2). Elle fait partiellement double emploi avec celle générale représen-

tée Fig. 124a et renforce la confiance en cette désactivation spécifique de valorisation.

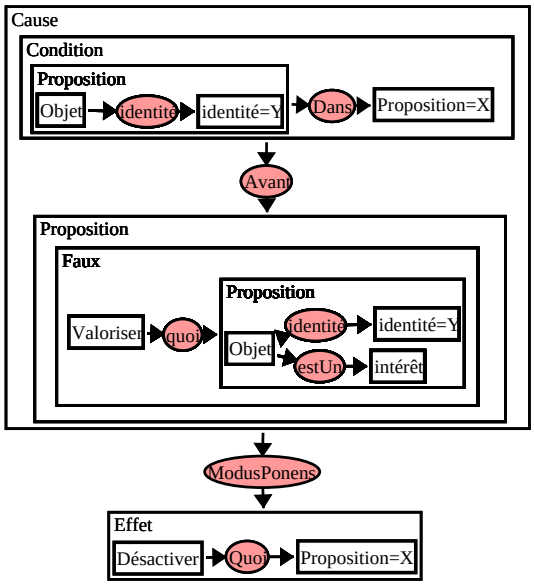
La désactivation effective dépendra de l'état actuel et de la confiance de cette volonté par rapport à l'impact d'une désactivation. Celles-ci peuvent être également modifiées comme les autres réactions par des cas : une désactivation affectant l'interaction voit son impact augmenter (sauf si l'impact initial est nul). Les outils ne disposent pas toujours d'une désactivation. Par exemple celle d'une tâche ou d'une technique d'interaction n'est pas définie. Des désactivations sont en fait gérées en interne puisque l'activation de ces adaptations est obtenues à l'issue d'un choix (désactivant les choix précédents).



(a) Volonté de désactivation générique



(b) Volonté de désactivation générique lié à un état anormal



(c) Volonté de désactivation des valorisations

FIGURE 124: Gestion de désactivations

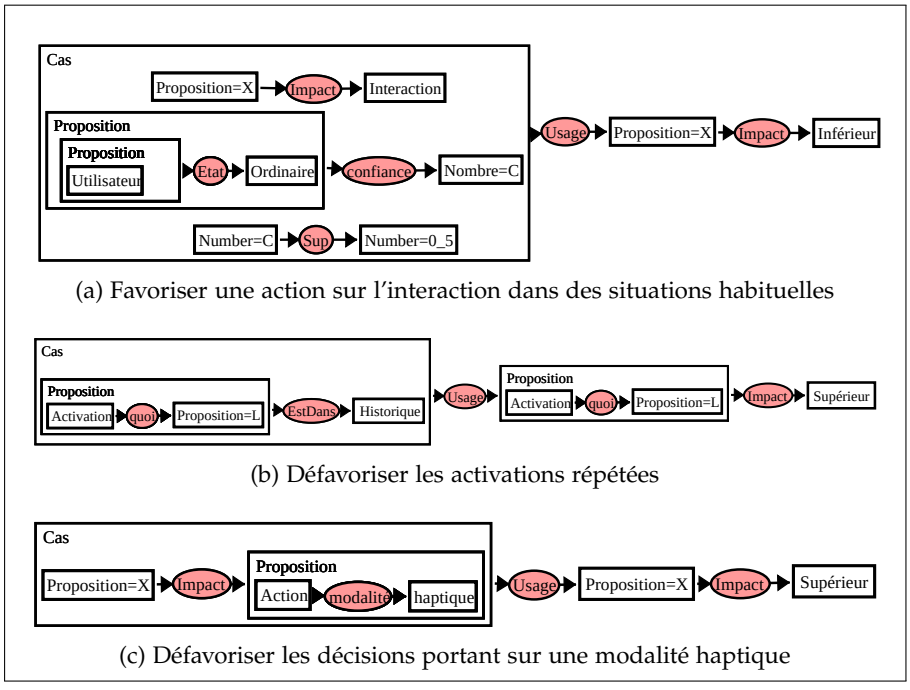
5.6.4.2 *Définition de cas*

Les *cas* sont des raisonnements particuliers du moteur affectant l'impact d'ensemble de décisions. Ainsi une réaction ayant un impact sur l'interaction est favorisée, via un impact diminué, dans le cas où les situations impliquant l'utilisateur semblent plutôt ordinaires (Fig. 125a). En effet, l'utilisateur peut profiter du changement en étant moins perturbé par celui-ci, lorsque son attention n'était pas alors déjà engagée outre mesure dans une tâche sortant de l'ordinaire.

Un deuxième cas permet au contraire de défavoriser une activation déjà présente dans l'historique (Fig. 125b). Ainsi une réaction spécifique qui a déjà été activée dans un passé proche et donc encore dans l'historique, ne peut être activée à nouveau que si elle a déjà été désactivée entre temps. Réactiver un effet qui a déjà été considéré comme pertinent à désactiver est ainsi plus risqué. Cela permet de plus d'ajouter une sécurité au moteur concernant les cycles d'activations et de désactivations. En effet, il est possible que dans une même situation, une réaction puisse être simultanément à activer et à désactiver (l'action finale est néanmoins conditionnée par l'état actif courant, section 4.5.2.1). Ainsi un objet peut être suffisamment intéressant pour obtenir une valorisation, mais pas suffisamment pour que celle-ci ne soit pas enlevée directement, notamment car les impacts de l'activation et la désactivation peuvent être différents.

Une autre sécurité est d'augmenter par défaut les décisions ayant un impact sur une modalité haptique (Fig. 125c). Car l'effet sur l'utilisateur est alors littéralement ressenti. Cela est probablement redondant, car l'impact initial de ces réactions devrait déjà être élevé. Mais il possible de définir un outil générique pouvant être à la fois visuel ou haptique (comme l'attraction présentée section 5.5.3). Si cet outil ne précise pas sa modalité, qui est unifiée en fonction des circonstances, il fournit alors un seul impact initial pour l'ensemble des modalités. C'est un choix de définition discutable mais envisageable. Ce cas permet alors à l'usage de la modalité haptique d'avoir plus d'impact que les autres, même dans le cas de cette définition générique.

Enfin des cas peuvent par la suite être également précisés pour tenir compte d'adaptations appréciées par l'utilisateur. Ce sont des préférences fortes car modifier l'impact permet de favoriser des adaptations par rapport à d'autres mais également d'en permettre une quantité simultanée plus importante. Jouer sur la confiance uniquement permet à l'utilisateur d'indiquer un usage privilégié sans en favoriser la quantité (comme la préférence d'une technique via la volonté de l'utiliser, section 5.4.4).



(a) Favoriser une action sur l'interaction dans des situations habituelles

(b) Défavoriser les activations répétées

(c) Défavoriser les décisions portant sur une modalité haptique

FIGURE 125: Cas modifiant l'impact de réactions en fonction des circonstances

5.6.4.3 *Règles de gestion des méta-adaptations*

Les règles de méta-adaptations actuelles sont très directes. Ainsi deux règles sont envisagées, reliant une information de fonctionnement (section 5.2.3) à l'usage d'un méta-effecteur (section 5.5.4). La configuration de réaction normale est utilisée lorsque les requêtes sont gérées de manière automatique, et le mode de réaction facile lorsque les requêtes sont déclenchées par l'utilisateur. Cela permet une gestion générale d'appréciation similaire aux *cas* : cela indique automatiquement que, pour une demande manuelle, des adaptations seraient appréciées, quel qu'elles soient. Une deuxième règle permet d'utiliser l'algorithme approché lorsque le temps entre deux réactions est élevé et l'algorithme précis autrement.

5.7 CONCLUSION SUR LA CRÉATION DE LA BASE DE CONNAISSANCES ET DES OUTILS

La création de processus intelligent, bien que nécessitant un cadre d'expression général, bénéficie d'une approche ascendante. Ainsi les nécessités, difficultés et les synthèses successives des réalisations précédentes font évoluer l'expression de ce processus. C'est pourquoi l'usage de sémantique facilement modifiable est intéressant. Des capteurs et des effecteurs ont été créés pour nos applications et ont également influencé la création du processus intelligent devant les gérer. Ces outils dont l'implémentation est certes dépendante du gestionnaire d'environnements virtuels, Virtools, ont été pensés comme généraux et réutilisables.

Premièrement des capteurs permettant d'extraire de la sémantique d'environnements virtuels classiques ont été réalisés. Par exemple, des attributs peuvent être exprimés pour divers objets. Une des informations sémantiques les plus intéressantes est l'intention de l'utilisateur. Des capteurs spécifiques permettent d'étudier son mouvement, de relever les objets proches de sa main, de se référer à l'historique, etc. afin de définir des sources d'intérêts. Chacun n'est seul qu'un faible indice mais la multiplication des sources permet au moteur de raffiner leur confiance en les fusionnant. Des objectifs peuvent être également estimés mais il est difficile d'être confiant autrement que par une communication directe avec l'utilisateur. Ainsi des capteurs permettant des commandes spécifiques (e.g. des gestes) peuvent être introduits pour les obtenir. De plus, les objectifs peuvent être connus et précisés par l'application.

La fusion des informations favorise globalement l'usage d'étapes dans le raisonnement, incluant mais non limitées aux intérêts et objectifs. Ces étapes rendent les différentes parties de raisonnement tant que possible indépendantes et aisément évolutives. Ainsi qualifier une situation d'ordinaire, de facile ou difficile, de particulière voire anormale, d'être un accomplissement ou de pouvoir exprimer des volontés d'actions sont autant d'étapes sémantiques. L'estimation des états et volontés peut être l'usage principal du moteur permettant l'analyse d'une situation.

Enfin, les effecteurs et les règles les déclenchant sont abordés. Des effecteurs permettant des valorisations variées d'objet, ainsi que de modifier l'interaction par le choix de technique ou de tâche sont créés. Afin de bénéficier de définitions et de réflexion séparées, des éléments d'adaptations sont alors introduits. Ceux-ci peuvent être combinés puis agrégés et mènent à des adaptations prenant en compte

les nombreuses possibilités en limitant les contraintes sur la description et la disponibilité des différents outils. Enfin, le raisonnement du moteur peut être influencé globalement. Il est ainsi possible de définir une même règle pour la désactivation des différentes mises en valeur ou de cas permettant de préciser les moments propices aux modifications de l'interaction.

Finalement le processus de réaction combine l'ensemble des informations afin de déclencher les outils les plus adaptés. La création de la base de connaissance et des outils détermine ainsi l'usage particulier qui est fait du moteur. Néanmoins la décomposition d'une situation inconnue en intérêt, objectif, état, volonté et éléments d'adaptation permet de définir des règles sémantiques aisément réutilisables. Ainsi cette base de connaissances permet l'usage du moteur représenté Fig. 126, qui correspond à l'illustration des étapes pratiques sur notre conceptualisation initiale (précédemment illustrée Fig. 50). Chaque étape peut être affectée directement permettant au raisonnement d'évoluer tout en tenant compte des capacités propres des applications et des préférences utilisateur. Les applications peuvent ainsi faire appel au moteur qui raisonne alors en fonction des situations à l'aide de cette base de connaissance et des outils effectivement présents dans chacune (ainsi que globalement des informations fournies dynamiquement ou non).

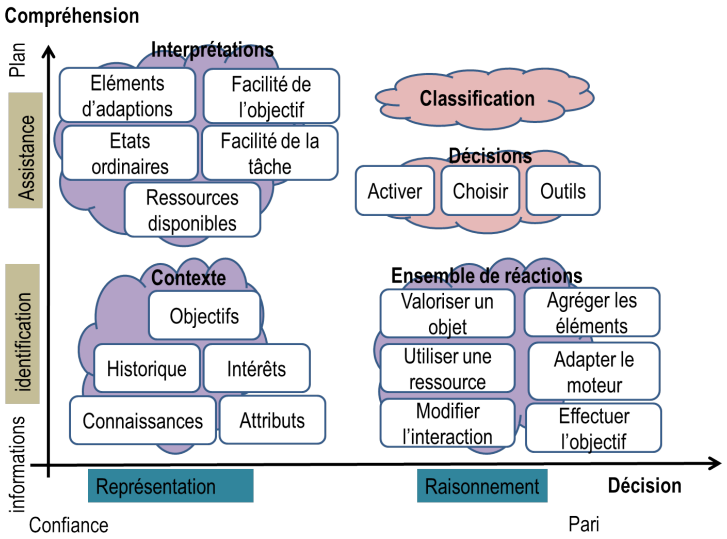


FIGURE 126: Étapes effectuées par la base de connaissance

6

APPLICATIONS ET COMPORTEMENTS D'ASSISTANCE OBTENUS

Résumé : *Après avoir créé des possibilités de raisonnement sémantique puis une série de règles et d'outils virtuels compatibles, le moteur est prêt à être utilisé. La première étape, effectuée en fait en parallèle des précédentes, a permis d'étudier les possibilités pratiques du moteur, via un premier scénario simple. Elle illustre les possibilités d'identification des intérêts et difficultés de l'utilisateur ainsi que des premières assistances perçues comme temps réel. Avec les améliorations du moteur, une application pour son évaluation a été envisagée. Cependant le temps de réponse rallongé peu à peu par un raisonnement étoffé rend désormais incompatible le moteur dans son ensemble avec un usage en ligne. Cette application permet néanmoins en indiquant l'objectif au moteur d'illustrer des comportements d'assistances en fonction de la situation relevée. Le moteur peut également être utile hors-ligne pour une analyse ou une conception d'activité. L'étude des étapes internes menant au choix de la technique d'interaction est alors illustrée et peut être appliquée pour une analyse de sessions enregistrées.*

6.1 PREMIÈRES APPLICATIONS

6.1.1 *Introduction*

Le moteur créé, utilisant la base de connaissance, est désormais apte à effectuer plusieurs formes de raisonnements. Notamment la demande de réaction automatique, processus parallèle demandé par les applications afin de gérer et déclencher leurs effecteurs, à partir d'informations supplémentaires dynamiques issues des capteurs. La base de connaissance est aisément extensible et apte à profiter de outils déjà créés. Des applications peuvent ainsi être envisagées, profitant du moteur pour obtenir, séparément ou simultanément, selon les besoins et l'expressivité utilisée :

- une assistance universelle en ligne automatique.
  L'application peut effectuer la demande de réactions génériques après chaque nouvelle information. Le moteur déclenche alors les effecteurs disponibles adéquats selon la situation. Chaque nouvel apport permet donc d'ajuster automatiquement les assistances quelles qu'elles soient.

- une assistance universelle en ligne manuelle.
  L'utilisateur peut également effectuer la demande de réactions, en sus du processus précédent ou en étant le seul à pouvoir réclamer assistance. Par exemple associer cette demande à un unique bouton le transforme en bouton universel

d'assistance.

- un détecteur en ligne automatique de situations particulières.
  L'application en ajoutant des effecteurs à impact nul dont les usages correspondent à des requêtes spécifiques, reçoit à l'issue du processus de réaction les solutions éventuelles à chaque requête. Par exemple, l'application peut, après chaque demande de réaction, être informée de l'estimation de l'ensemble des intérêts de l'utilisateur et de leur confiance ou de la difficulté de la situation actuelle.

- une demande d'interprétation en ligne explicite.
  Similaire au cas précédent mais les requêtes spécifiques peuvent ne pas être intégrées à un effecteur géré par le processus de réaction générique. Elles peuvent être exprimées seules directement, à tout moment :
  - par l'application qui forme dynamiquement ou envoie une requête préformée puis réagit ou stocke la réponse.
  - par le concepteur qui exprime une requête pendant le déroulement de l'application lui permettant de suivre ses comportements ajoutés et leur pertinence. Cela permet de préparer un mode automatique via le moteur ou d'obtenir une aide à la conception d'environnement virtuel.
  - par l'utilisateur via une interface spécifique. Celle-ci, in fine, pourra être une interprétation ontologique d'une phrase simple. Mais il est déjà possible de définir un ensemble de requêtes préformées déclenchées par des commandes spécifiques. Par exemple définir des conseils tels que : Quelle technique utiliser ? Quels sont les attributs de l'objectif ?

- une demande d'interprétation en ligne implicite.
  La requête spécifique précédente peut être un ajout d'information. Dans le cas de l'utilisateur cela peut être la description de son profil hors ligne. Mais un ajout d'information en ligne dénote d'une volonté d'interprétation implicite. Par exemple, l'utilisateur exprimant un objectif, via une interface spécifique, peut avec le mode automatique obtenir directement une assistance pour celui-ci.

- une analyse hors ligne de l'activité utilisateur.
  En enregistrant les différents évènements capteurs issus d'une session, le concepteur ou l'application peut par la suite analyser le comportement de l'utilisateur hors-ligne en détectant automatiquement des caractéristiques, comme l'apparition d'intérêts ou de difficultés (selon les effecteurs ajoutés contenant les requêtes automatiques). Les demandes d'interprétations peuvent être explicites tout au long du déroulement de la session hors-ligne récréée.

Le moteur est sémantique, facile à connecter et à modifier ainsi qu'extensible. Chaque règle peut être ou non chargée indépendamment des autres (à l'exception du noyau du moteur qui définit le méta-interpréteur, lui-même séparé en un fichier de gestion de confiance, de temps et de processus de décision inter-dépendants). Ainsi les différents concepts, le choix, l'activation, la fusion, les différentes situations et éléments d'adaptation etc. peuvent être utilisés ou non.

6.1.2 *Scénario*

Un premier scénario a permis de tester les différentes évolutions du moteur jusqu'à la multiplication des effecteurs. Dans ce scénario, l'accent a été porté sur la détection d'intérêts, le premier palier pour une assistance complètement automatique. Les assistances sont moins élaborées et permettent principalement la mise en valeur de la détection des intérêts et des difficultés. Ainsi cette application met en scène un environnement virtuel simple (Fig. 131). Le moteur doit pouvoir automatiquement détecter les intérêts de l'utilisateur, notamment reliés à sa main [1]. La main virtuelle est directement liée au pointeur de Fly over qui est la technique d'interaction utilisée. Les autres outils disponibles pour cette application sont indiqués Fig 128. Un des avantages du moteur est d'ailleurs la gestion des outils présents ou non. Les règles utilisées sont un sous-ensemble de celles présentées précédemment, parfois sous une forme quelque peu différente. Parmi les différences majeures, les attributs d'objets ne sont pas relevés et les adaptions ne sont pas décrites comme la combinaison d'*éléments* d'*adaptations* (tributaire des attributs notamment) tels que définis section 5.6.2.1. Le processus combine différentes règles et situations dont l'ensemble des valorisations qui sont envisageables en permanence. L'application ajoute également au démarrage des règles et des faits propres en plus de la définition des outils disponibles. La surveillance spécifique de la main est ajoutée, ainsi que des règles reliant les outils de l'applications. Une partie de ces règles permettent une commande directe du moteur à travers l'interprétation des gestes. Finalement, les règles disponibles sont résumées Fig.129.

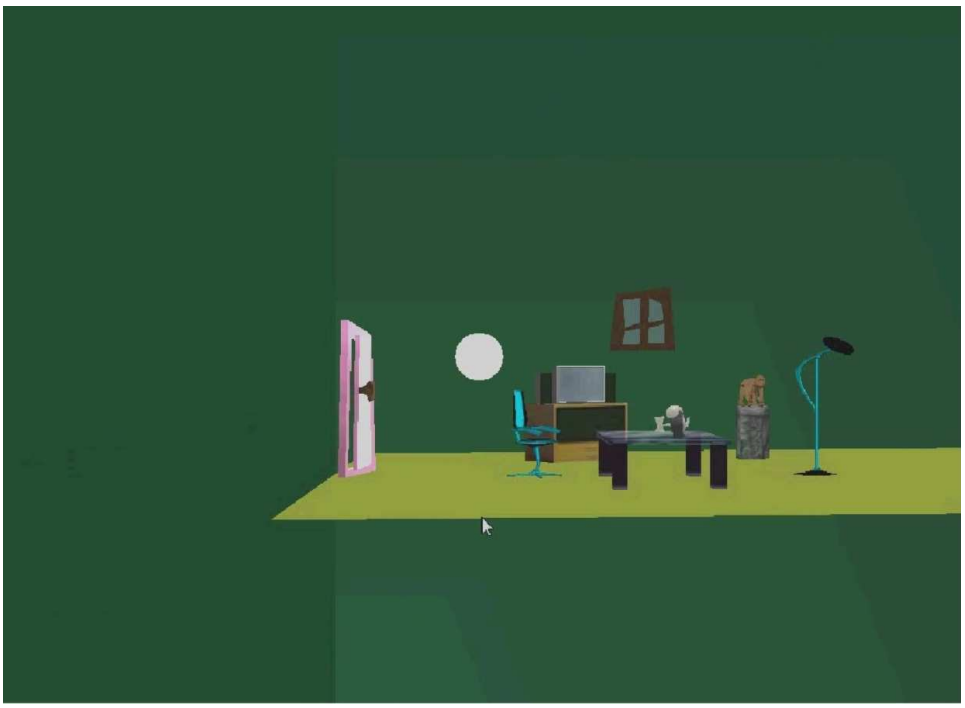


FIGURE 127: Environnement virtuel de test : objets de complexité diverse et Fly-over pour l'interaction. La sphère blanche est associée à la main de l'utilisateur

1. Le nom de l'objet sous Virtools et ainsi associé à l'utilisateur par l'ontologie du moteur.

Outil	Nom	Buts	Sections
Capteur	Zones d'intérêt	Reporter les objets d'une zone 3D créée comme intéret	5.3.2.1
	FlyOver	Reporter les objets sous le pointeur de flyover en tant qu'intérêt	5.3.2.2
		Reporter les objets manipulés	
		Reporter la tâche courante	
	Attributs mvt	Qualifier un mouvement : sa portée et sa vitesse	5.3.4.1
	Gestes	Détecter un geste d'un alphabet et récupérer ses attributs	5.3.4.2
	Types demandes	Reporter le type de demande de réaction	5.2.3.3
Effecteur	Colorisation	Modifier la couleur d'un objet	5.5.2
	Attraction	Ajouter une force haptique ou visuelle sur l'objet	5.5.3
	Zoom	Centrer la caméra ou zoomer sur un objet	5.5.3
	Choix de tâche	Ajouter un objet à la sélection active	5.5.3
Méta-effecteur	Config Moteur	Modifier les paramètres du moteur	5.5.4

FIGURE 128: Outils disponibles pour la première application

Type	Règles	Sections
Intrinsèque au moteur	Gestion des informations selon leur sémantique	4.2.3 4.3.3.1
	Influence du temps sur la vérité et la confiance	4.3.2.2
	Fusion des informations multiples pour la confiance	4.3.3.2 4.3.3.3
Obtention d'intérêts	Issus d'une zone d'intérêts activée	5.3.2.1
	Issus du pointeur de Fly-over	5.3.2.2
	Issus d'un intérêt précédent dans l'historique	5.3.3
Obtention d'objectifs	Un intérêt peut être un objectif	5.3.5
	Un objectif précis peut être la tâche courante liée à un objet objectif	5.3.5
Contrôle des valorisations	Valoriser les intérêts et objectifs	5.4.2
	Valorisation possibles : colorisation, attraction ou zoom	5.6.1
Raccourcis de tâches	Sélectionner automatiquement un objet pour un objectif de sélection	5.4.2
	Se déplacer automatiquement vers un objet pour un objectif de navigation	5.6.1
Gestion des	Lien entre la volonté d'activer et la possible activation	4.5.2.1
	Retirer une modification de l'interaction active anormale	5.6.4.1

Type	Règles	Sections
activations	Retirer une modification visuelle active si la valorisation est accomplie	5.6.1
Influences globales	Augmenter l'impact des réactions usant de modalité haptique ou modifiant l'interaction	5.6.4.2
	Augmenter l'impact d'une décision déjà effectuée et encore présente dans l'historique	5.6.4.2
	Diminuer l'impact d'une modification de l'interaction pour un mouvement local	5.6.4.2
Meta-adaptations	Faciliter les réactions pour les demandes manuelles	5.6.4.3
Spécifiques à l'application	Surveillance des mouvements de la main de l'utilisateur	5.3.4.1 5.3.4.2
	Questionner et obtenir les attributs d'un geste détecté	5.3.5
	Activer une zone d'intérêt dans la direction d'un geste, dans le cas d'un geste rectiligne (possédant plusieurs sous-types)	lié à 5.3.2.1
	Activer une zone d'intérêt autour de la "main" à l'issue de la détection d'un geste circulaire	lié à 5.3.2.1
	Désactiver les zones d'intérêt lié à un direction après 3s	lié à 5.3.2.1
	Désactiver l'ensemble des adaptations active lors de la détection d'un geste "Z"	lié à 5.6.4.1

TABLE 129: Comportements et règles inclus au premier scénario

6.1.3 *Comportements obtenus pour l'application*

6.1.3.1 *Illustration des résultats*

Au final, les règles se combinent comme on pouvait s'y attendre, avec cependant des résultats supplémentaires qui n'étaient pas pleinement prévus. Ces résultats dépendent des impacts initiaux, de l'impact total admissible et des confiances des règles et faits. Une part des adaptations obtenues est illustrée Fig. 130. Un des avantages du moteur est de pas avoir à expliciter l'ensemble des transitions effectuées et des comportements désirés.

6.1.3.2 *Détection effective des intérêts et objectifs*

En pratique, les intérêts peuvent ainsi être relevés sous différentes formes. Ils peuvent être premièrement explicites. Une zone d'intérêt peut être créée volontairement (et est visuellement affichée) autour de la main par un geste circulaire afin que les objets proches soit détectés. De même, il est possible de pointer directement un objectif, par les zones créées à l'aide de gestes rectilignes. Le capteur gérant la reconnaissance de geste le fait à partir de la position du pointeur virtuel de Fly-over (et non pas sur la position 3D réelle de la main utilisateur) et provoque ainsi principalement des détections pendant la phase de sélection fine gérée par Fly-over, lorsque la caméra est fixe et les mouvements virtuels proches des mou-

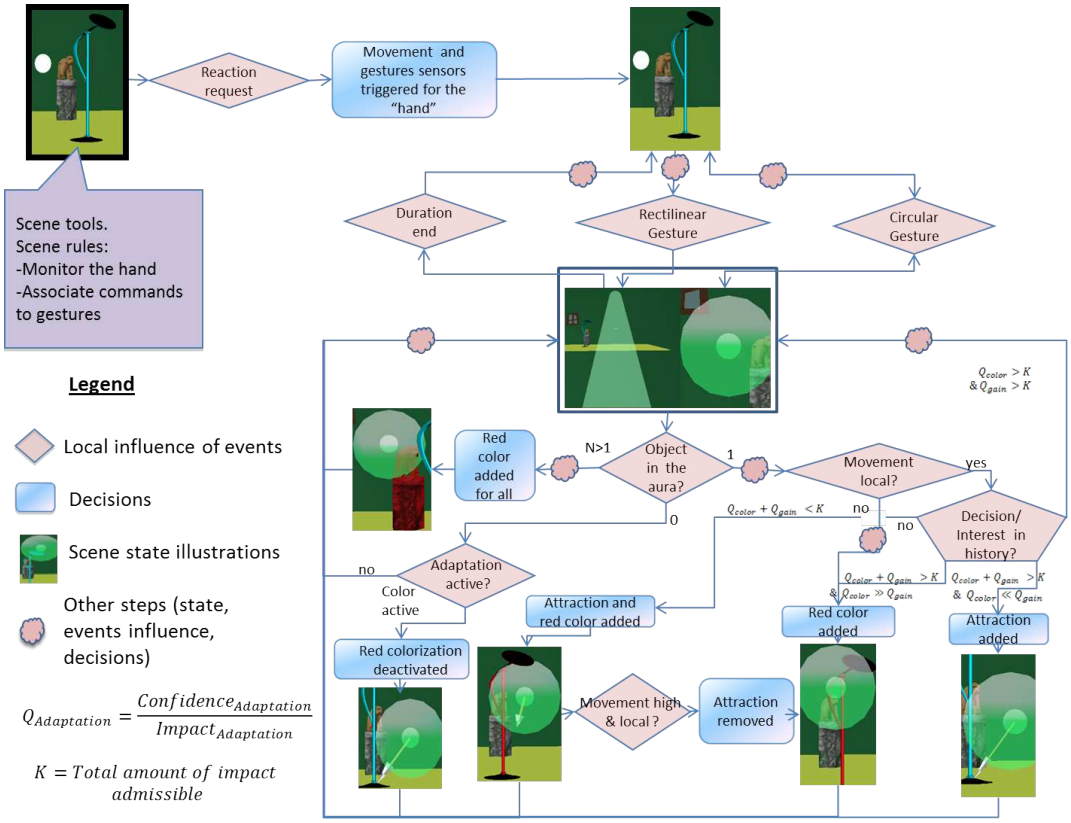


FIGURE 130: Illustration d'un partie des adaptations automatiques dépendant du contexte

vement réels. Enfin centrer et laisser volontairement le pointeur de Fly-over au dessus d'un objet l'indique comme un intérêt qui se renforce avec le temps. Deuxièmement, les intérêts peuvent également être implicites. Les intérêts d'un passé proche, encore présents dans l'historique, sont notamment considérés. De plus passer au dessus d'un objet avec le pointeur, par exemple pendant une réflexion, permet de le relever comme intérêt potentiel, lorsque le survol est suffisamment lent. Enfin un intérêt peut être détecté à l'issue d'un déplacement précis et rectiligne en direction d'un objet. En effet dans ce dernier cas, la reconnaissance de geste également active pendant la navigation, peut provoquer la création de zone. Dû à la différence d'apprentissage et au fait que le déplacement virtuel soit effectué après un traitement de la position réelle, seuls des gestes rectilignes sont détectés, dans les cas où la navigation est très directive. La création d'une zone d'intérêt due à un mouvement circulaire, issue donc de la navigation en tournoyant autour d'une position est également théoriquement possible mais ne se produit pas en pratique avec la base de gestes considérée. Au final, la combinaison et la fusion de l'ensemble des informations de ces sources différentes permettent ainsi la détection des intérêts principaux possédant une confiance suffisante et pouvant alors être mis en valeur. Les objectifs peuvent être également mis en valeur mais sont issues uniquement d' une interprétation possible des intérêts eux-même. Ils ont également leurs propres adaptations.

6.1.3.3 *Adaptations réactives et valorisation*

INFLUENCE DU MOUVEMENT DANS LE CAS D'UN INTÉRÊT UNIQUE Avec une zone d'intérêt active autour de la main, passer près d'un objet suffit pour obtenir la confiance nécessaire à sa colorisation. Sans plus de contexte l'attraction n'est pas

ajoutée car elle a un impact initial plus élevé et le moteur est plus prudent vis-à-vis des modifications d'interaction. Cependant, rester près d'un objet permet de le rendre attractif. En effet, le mouvement de l'utilisateur étant alors local, l'impact des réactions modifiant l'interaction est réduit. La modification des couleurs est désactivée lorsque l'utilisateur s'éloigne de l'objet valorisé sa confiance diminuant alors rapidement. L'attraction est retirée lorsque l'utilisateur lui résiste. L'attraction visuelle profite des deux règles mais sa désactivation possède un impact plus important et nécessite ainsi un apport supplémentaire. La seule persistance de l'intérêt passé permet autrement de conserver l'attraction active malgré la confiance diminuée. Les valorisations sont également désactivées quand l'objet est sélectionné.

En pointant un objet ou en se déplaçant globalement en ligne droite vers un objet, celui-ci est alors mis en rouge à grande distance. Ce qui fut plus surprenant c'est que pointer un objet ou se déplacer vers lui directement en partant d'un position de repos, ajoute l'attraction à l'objet également. En effet, la direction issue de cette intention première peut être distinguée des autres par l'effet de latence du capteur gérant le relevé des attributs du mouvement : le mouvement est alors encore considéré comme étant local, ce qui réduit l'impact d'un modification affectant l'interaction. Bien que non prévu initialement ce comportement permet de prendre plus sérieusement les intérêts issus de cette phase de réflexion et d'observation où l'utilisateur était quasi-immobile. Les couleurs modifiées ainsi à distance sont remises à zéro au bout de quelques secondes.

INFLUENCE DU TEMPS Dans les deux cas précédents, lorsque des objets sont relevés plusieurs fois comme étant des intérêts potentiels, l'attraction peut être activée quel que soit le type de mouvement (local ou global). En effet, ces intérêts sont alors présents plusieurs fois dans la base de connaissance et/ou dans l'historique augmentant leur confiance à l'issue de la fusion des informations. Lorsque l'attraction est désactivée, elle n'est alors généralement pas réactivée pour un laps de temps correspondant à la mémoire de l'historique. En effet les impacts d'activations répétées sont augmentés et l'impact initial de l'attraction est plus élevé. La colorisation peut, elle, être réactivée quelques fois. De plus selon l'impact initial de celle-ci et de l'historique, le temps de la modification de couleur peut varier et peut également scintiller pendant quelque temps, ce qui n'était pas prévu de prime abord. En effet, un objet peut être simultanément suffisamment intéressant pour déclencher la colorisation mais néanmoins pas assez pour éviter une désactivation de l'adaptation quasi-immédiate. La règle augmentant l'impact des activations successives a d'ailleurs été introduite pour répondre à ce souci et ramener un état stable à la longue. L'ensemble résulte en un moyen d'attirer l'attention, initialement non prévu. Un objet qui a été suffisamment intéressant et qui semble alors abandonné par l'utilisateur pendant un moment se met alors à scintiller. Cela dure jusqu'à ce que l'utilisateur le reconsidère, ré-augmentant sa confiance et favorisant la colorisation, ou jusqu'à ce que l'intérêt de l'objet délaissé atteigne un niveau suffisamment bas avec le temps.

CAS D'INTÉRÊTS MULTIPLES Des situations plus complexes surviennent lorsque plusieurs objets sont simultanément des intérêts. Dans le cas de confiances variées, les réactions sont toutes aussi variées. Avec des confiances similaires en revanche, par exemple pour plusieurs objets proches avec une zone d'intérêt activée, une tendance apparait. Souvent, seule l'adaptation avec le moins d'impact, la colorisation, est appliquée à un maximum d'objets. Le moteur en soi n'a pas de traitement

spécifique pour les groupes d'objet, néanmoins dans un contexte similaire pour chacun, la solution la plus adaptée est souvent l'adaptation la moins couteuse, appliquée au plus d'objets. Il n'y a alors rapidement plus assez d'impact disponible. Ainsi une logique de groupe apparait, bien que non explicitement décrite.

6.1.3.4 *Adaptations pro-actives : raccourcis pour l'accomplissement d'objectif*

Connaissant la tâche courante, des objets d'intérêts peuvent être interprétés comme faisant partie d'un objectif potentiel. Ainsi dans le cas d'une tâche de sélection, déclencher la manipulation automatique d'un objet peut se produire. De même, effectuer un zoom, en déplaçant la caméra utilisateur, effectue partiellement un objectif de navigation et est une adaptation possible durant cette même tâche. Ces adaptations sont pro-actives puisqu'elles supposent ainsi d'effectuer une partie de la tâche, reconnue automatiquement, avant que celle-ci ne soit effectivement terminée par l'utilisateur. Ces décisions sont rares car de nombreuses règles sont impliquées notamment celle déterminant un objectif à partir d'intérêt, de faible confiance. De plus, ces décisions ont un impact élevé.

6.1.3.5 *Méta-adaptations*

Le moyen le plus facile pour obtenir des adaptations pro-actives comme présentées précédemment est de passer en mode manuel. Cela permet alors à un méta-effecteur de modifier dynamiquement les paramètres du moteur afin d'obtenir plus de réactions, à la fois leur nombre simultané mais aussi la facilité de déclenchement de chacune. Ainsi les décisions difficiles à prendre, concernant un objectif hypothétique, peuvent se produire même si la confiance d'objet ciblé n'est que moyenne. En effet, si l'utilisateur est la source de la demande d'adaptation, le besoin d'assistance est avéré. Il est également plus probable que l'utilisateur ait un objectif en tête. La réduction des spécifications préalables (sur l'impact et la pertinence) permet ainsi de favoriser l'apparition d'assistances notamment celles ayant précédemment trop d'impact, comme les raccourcis de tâche.

6.1.3.6 *Temps de réponse*

Le moteur possédait ainsi un jeu de règles et d'outils moins élaboré, et le processus de réactions était alors effectué entre 200ms et 2s. La moyenne des réactions était légèrement inférieure à la seconde. Cela ne correspond pas à du temps réel, avec un seuil de perception visuelle de l'ordre de 200ms. Néanmoins lorsque le processus de réaction est automatique, il est également transparent pour l'utilisateur, qui n'a donc pas connaissance du début du cycle de réaction. Le décalage est alors rarement perceptible : des réactions pertinentes vis-à-vis du contexte en cours sont bien déclenchées et semblent temps réel. Dans le cas d'une demande manuelle, 2s avant réaction, bien que ressentie, sont acceptables avant d'obtenir une assistance universelle qui est notre but.

6.1.4 *Évolution des comportements et résultats intermédiaires*

L'évolution des nécessités menant à de nouveaux concepts et expressions ont déjà été abordées, notamment pour la notion d'activation, de choix, de fusion. Certains comportements ne sont pas toujours aussi facilement prévisibles et leur apparition

au cours des tests d'application (et autres résultats intermédiaires) a contribué à l'élaboration des règles moteurs.

6.1.4.1 *Conséquence du choix de l'inscription de faits ou d'évènements*

Les capteurs peuvent inscrire des faits ou des évènements. Les évènements sont propices au capteurs qui ne maitrisent pas forcément la durée de leur information, comme la reconnaissance de gestes. Mais cela reste un choix d'implémentation pour de nombreux capteurs, qui influence le comportement global. Par exemple avec une confiance moyenne pour les intérêt introduits par le capteur de zone, on peut voir apparaitre une alternance d'activations et de désactivations de la colorisation. Ce phénomène peut être différent selon l'implémentation choisie. En effet sans la règle sur la maitrise des activations répétées :

- le capteur peut inscrire des faits pour les objets de la zone et les retirer lorsque ceux-ci n'y sont plus : le clignotement est alors continu tant que l'utilisateur est proche.
- Le capteur peut inscrire un unique évènement : le clignotement se produit et la couleur initiale retrouvée, même si l'utilisateur reste constamment proche de l'objet.
- Le capteur peut inscrire régulièrement des évènements : le clignotement possède un régime transitoire assez irrégulier. Mais en restant proche de l'objet, l'état final est la colorisation alors qu' en restant loin, c'est le retour à la couleur initiale.

L'implémentation finale est celle utilisant les faits, mais avec l'augmentation de l'impact des activations répétées.

6.1.4.2 *Introduction de la règle sur les activations répétées*

C'est bien l'activation répétée, impliquant une désactivation répétée, qui doit déclencher une augmentation de son impact. Elle permet de gérer par exemple correctement le cas d'un objet dont la confiance est suffisante pour l'activation de la colorisation mais insuffisante pour sa désactivation quasi-immédiate. Le clignotement est alors uniquement temporaire et se termine toujours par la couleur originale de l'objet. Considérer comme premièrement l'ensemble des répétitions de décisions comme suspectes n'est pas assez précis : c'est non pas la répétition seule mais l'alternance de décisions contraires qui n'est pas désiré. De plus, notre règle sur l'impact des activations répétées favorise la désactivation. Autrement le clignotement de l'objet est maitrisé mais l'état final peut être celui de la couleur modifiée, de l'état actif.

6.1.4.3 *Introduction de la règle sur les intérêts passés*

La règle permettant de gérer un intérêt passé dans l'historique comme étant également partiellement un intérêt courant a été ajoutée afin de fournir au moteur un mécanisme général pour conforter l'intérêt d'un objet considéré plusieurs fois. En effet si le comportement de l'utilisateur semble insister sur l'importance d'un objet, ajoutant plusieurs faits dans la base de connaissance, la fusion d'information permet d'obtenir une meilleure confiance. Cependant cela n'était possible que dans le cas d'une gestion par l'outil d'événements. Car si celui-ci gère mieux ses informations, avec la détection d'un changement d'état, ajoutant ou retirant un unique

fait dans la base de connaissance, l'effet d'accumulation n'était plus présent.

Cela permet également en pratique de contrebalancer l'augmentation d'impact d'une activation répétée. Celle-ci est nécessaire afin de ne pas ré-ajouter une force dérangeant l'utilisateur et plus généralement pour la stabilité des décisions. Cependant la reproductibilité du raisonnement peut être altérée. Revenir vers un même objet dans la même situation peut ne pas provoquer la même réaction. Le mécanisme des intérêts issus de l'historique permet de débloquer à nouveau une activation sur insistance de l'utilisateur.

6.1.4.4 *Évolution de l'historique*

La mauvaise interprétation de la notion de confiance d'un fait dans l'historique, avec une confiance des faits initiaux non prise en compte, a pu être détectée et modifiée suite à l'apparition de comportements étranges. A l'origine sans traitement particulier, l'information exprimant une situation comme présente dans l'historique est soit vraie, soit fausse, selon que l'accès à l'information dans cette zone à part est ou non possible. Mais avec la règle introduisant les intérêts passés comme étant potentiellement actuels, un objet dont l'intérêt était estimé faible par un évènement pouvait voir sa confiance augmenter une fois passé dans l'historique. Ainsi lorsque la première série d'évènements issus de la zone d'intérêt autour de la main se périmaient, la confiance augmentait brusquement et permettait des adaptations à fort impact, pas toujours bienvenues.

De même l'historique a été modifié pour associer à l'archivage d'un fait une date correspondant au moment présent, transformant le fait en évènement, dont la suppression finale est automatiquement gérée. Les faits passés dans l'historique n'avaient au départ pas de date associée et y restaient alors en permanence. Les évènements seuls bénéficiaient d'une péremption dans l'historique. En effet, la suppression de faits par défaut était définitive et il fallait indiquer explicitement leur archivage. Dès lors que les capteurs contrôlaient mieux leur propre information, les supprimant de la base de connaissance eux-mêmes, il était intéressant pour connaitre les intérêts passés de modifier cette suppression par défaut. En l'état, les faits était donc stockés dans l'historique, provoquant un ralentissement notable de l'application à la longue et une confiance sans cesse grandissante pour les intérêts. Les adaptations étaient alors de plus en plus lentes et de plus en plus osées. Jusqu'a ce que, l'impact des activations répétés augmentant également, plus aucune décision ne soit prise puisque toutes dépassaient l'impact total admissible.

6.1.4.5 *Correction automatique de bugs*

L'intérêt premier du raisonnement sémantique est de pouvoir être compréhensif et donc également réutilisable, modifiable, combinable etc. Ainsi décrire de cette manière le raisonnement permet d'avoir de bons comportements également dans des cas non prévus. Et par cela, nous pensions principalement à de nouvelles situations dans un environnement virtuel fonctionnel. Mais parfois un souci dans l'environnement virtuel peut être ainsi détecté par le moteur et des adaptations appliquées peuvent même résoudre partiellement le problème. Ainsi les outils sont créés dans Virtools indépendamment. L'utilisation commune de l'attraction et de fly-over a initialement posé problème. Il ne peuvent tous les deux imposer la position du pointeur. Il fallait indiquer à l'effecteur d'attraction la commande du pointeur

de Fly-over pour qu'il la modifie. En considérant la mauvaise variable, qui n'est alors qu'une étape dans l'élaboration de la position du pointeur virtuel par Fly-over, des modifications brusques de cette variable pouvaient se produire. Cela impliquait la perception de téléportations par l'attraction, qui était alors modifiée elle aussi brusquement. Le comportement autour de l'objet attractif devenait chaotique. Le mouvement étant alors anormal pour le moteur en présence d'une attraction active, celle-ci était désactivée. Bien qu'initialement pensée comme désactivation pro-active vis-à-vis d'un difficulté utilisateur, le raisonnement général est ainsi également appliqué et permet d'éviter l'incompatibilité des outils, en désactivant celui qui peut l'être.

6.1.4.6 *Introduction de gestes et limitations*

Initialement la surveillance de la main était automatique : la zone d'intérêt autour de la main était activée par défaut tout comme le relevé de ses attributs de mouvements. Puis les gestes ont été gérés afin de permettre le contrôle du moteur par l'utilisateur. Activer la zone d'intérêt autour de la main est alors déclenché par un geste circulaire. Il a suffit de transformer la surveillance de la main afin qu'elle soit un effet d'un geste circulaire déclenché. Le geste "z" permettant de remettre à zéro les adaptations déclenche alors la volonté de désactivation de toutes adaptations actuellement à l'état actif. Enfin un geste rectiligne a été associé à une nouvelle zone, directionnelle, exploitant une des commandes possibles de l'outil gérant leur création. Afin d'obtenir la direction, un capteur a été ajouté et une règle associée afin de le déclencher. Cependant il faut alors deux cycles de réactions pour créer cette zone. Le premier pour décider de la question de direction du geste et le second pour créer la zone en fonction de la réponse à celle-ci. Ainsi il peut s'écouler 4 secondes dans cette configuration avant l'apparition de la zone. Celle-ci n'est alors non seulement plus pertinente mais parfois décalée car définie relativement à la main de l'utilisateur, qui a continué son mouvement.

Ainsi en choisissant l'expressivité, la rapidité d'exécution est défavorisée comme on s'y attendait. Cela illustre ainsi une première limitation. Il est en effet préférable de ne pas imposer le moteur au sein de processus nécessitant le temps réel (un exemple extrême est le contrôle fin d'un retour haptique) mais comme superviseur. Le moteur peut être utilisé pour les contrôler globalement, par exemple les activer, sans pour autant devenir un intermédiaire nécessaire. Ainsi, pour être fonctionnel au vu du temps de réponse, le geste rectiligne doit déclencher directement la création de zone sous Virtools. La combinaison des deux modules Virtools peut être alors déclarée comme un nouvel outil pour le moteur, capable de surveillance d'un objet, notamment de la direction de son mouvement. De manière similaire Fly-over est ainsi gardé indépendant, même si le moteur obtient des informations de son fonctionnement interne et qu'il peut également partiellement le contrôler.

6.1.5 *Conclusion*

En utilisant le moteur, il a ainsi été possible d'obtenir automatiquement des adaptations en fonction du contexte. Le moteur s'occupe des assistances adéquates tout comme de l'identification des situations. Par exemple, les intérêts sont également détectés automatiquement et peuvent être extrapolés comme étant des objectifs selon la tâche. Un objectif peut ainsi provoquer une adaptation pro-active. Des

comportements non prévus initialement, mais cohérents et adéquats, surviennent. Cela permet d'illustrer la gestion multiple de situations que l'on peut obtenir par la définition des règles élémentaires sémantiques. Les réactions du moteur peuvent apparaitre temps réel pour l'utilisateur, qui n'a pas conscience du début du processus, en mode automatique. Demander manuellement des adaptations permet au moteur de se paramétrer automatiquement afin d'être plus ambitieux dans ses réactions. L'attente est alors perceptible, quelques secondes au plus, mais est acceptable. En effet, elle permet de répondre aux besoins d'un grand nombre de situations à partir de la même commande, d'un unique bouton d'assistance universelle, ce qui est notre but recherché (tout en maintenant l'accès et l'évolutivité des étapes de l'approche).

6.2 ILLUSTRATION DES ASSISTANCES DANS UN SCÉNARIO D'ÉVALUATION

6.2.1 *Scénario*

Ce deuxième scénario met en oeuvre des situations différentes pour six objets cibles (Fig.131). Le moteur devrait ainsi déclencher des adaptations bien différentes dans chacun des cas. Cet environnement virtuel est instrumenté et scripté afin de permettre l'évaluation du moteur. L'utilisateur doit ainsi récupérer parmi les cibles potentielles celle de couleur différente et la ramener au point de départ. Après chaque tâche réussie le scénario est redémarré et une autre cible est aléatoirement choisie parmi celles n'ayant pas encore été testées.

Les capteurs d'attributs et les éléments d'adaptations ont été introduits pour ce scénario, ainsi que l'agrégation. Une gestion souple des attributs a aussi induit la notion de contraire. Les règles sont ainsi globalement les précédentes (Fig. 129) plus celles présentées Fig. 133. L'expression de chaque règle est également globalement plus proche de la forme présentée dans le manuscrit (avec introduction des étapes supplémentaires dans certaines règles précédentes, comme la notion d'*accompli* pour la valorisation, la description des outils etc.). Néanmoins, le temps de réactions augmente fortement, pouvant atteindre 2 min. Une tâche impliquant l'utilisateur avec de tels délais n'est pas envisageable. L'usage des éléments d'adaptations sans l'agrégation (dans des cas maitrisés, section 5.6.2.3) et la réduction des outils disponibles permettent de réduire ce délai. Les outils utilisés sont présenté Fig.132. L'attente pour un cycle de réactions est alors de 5 à 20s , ce qui est encore trop élevé. Ce scénario initialement prévu pour évaluer le moteur permet néanmoins l'illustration de la variété et de la souplesse des assistances en fonction du contexte. C'est un cas quelque peu différent du précédent puisque même si la détection d'intérêt peut être également activée, le moteur peut ici être informé à l'avance de l'objectif courant.

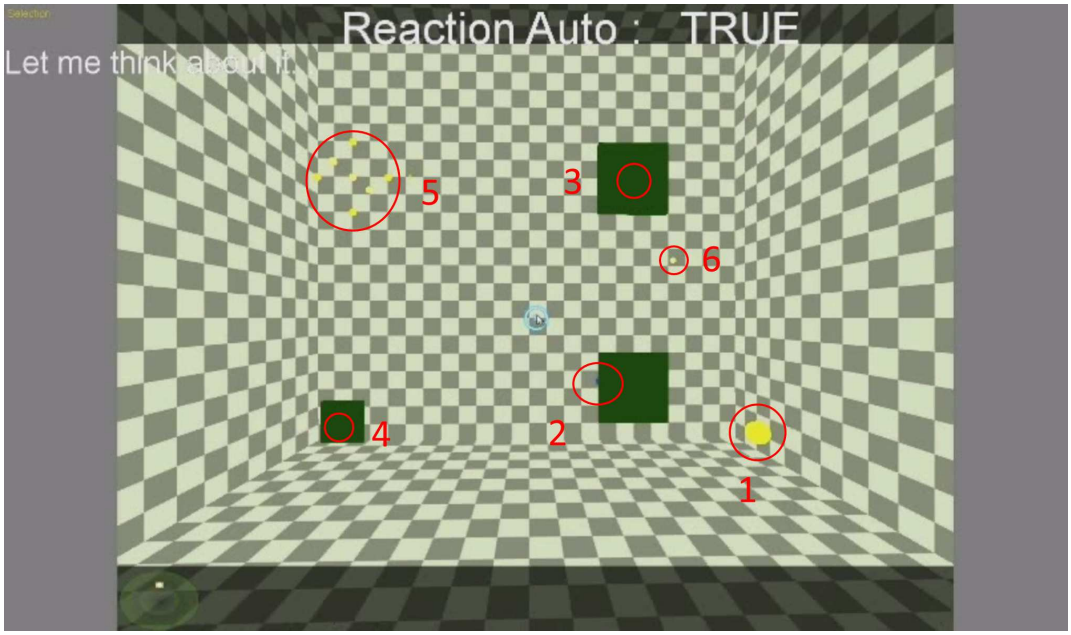


FIGURE 131: Environnement virtuel d'évaluation : six situations d'objets aux attributs différents, certains cachés, pour illustrer l'assistance. L'état de processus de décison (en attente, en réflexion ou déclenchant des assistance) et le mode automatique ou non sont affiché en haut de l'écran.

Outil	Nom	Buts	Sections
Capteur	Attributs	Ensemble d'attributs relevables	5.2.1
Effecteur	Pulsation	Cycle d'agrandissement et de réinitialisation de la taille d'un objet	5.5.2
	Trajectoire	Afficher une trajectoire vers une cible par des particules	5.5.2
	Bip	Indiquer la distance par la fréquence d'occurence d'un son	5.5.2

FIGURE 132: Outils disponibles pour la deuxième application

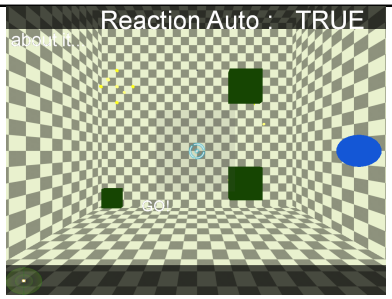
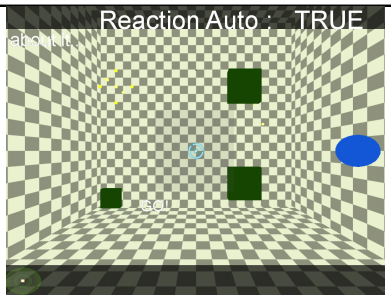
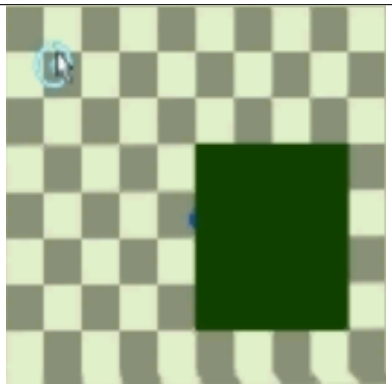
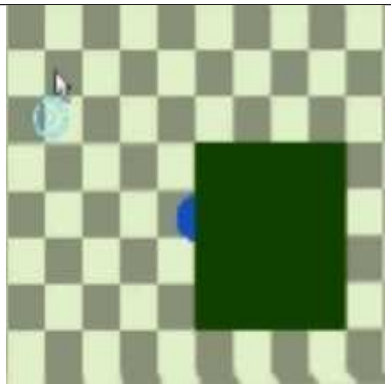
Type	Règles	Sections
Contraires	gestion générale des contraires notamment pour les attributs	4.5.1.2
Contrôle des valorisations	Élément d'adaptations	5.6.2.1
	Activation d'une adaptation combinée possible	5.6.4.1
Spécifiques	Définition de l'objectif courant par l'application	

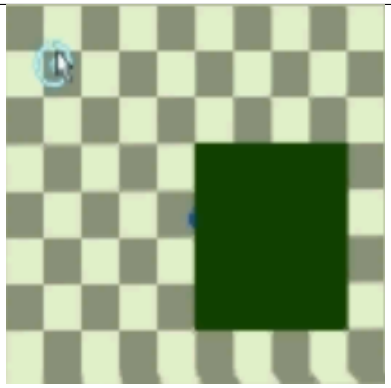
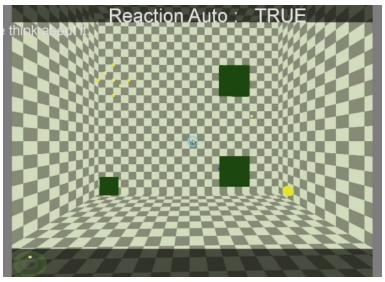
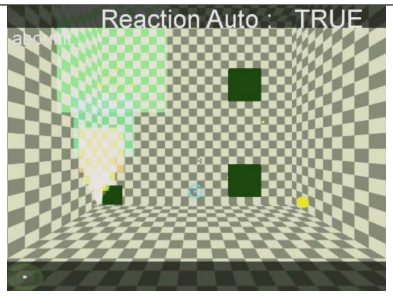
FIGURE 133: Comportements et règles supplémentaires pour le deuxième scénario

6.2.2 *Comportements obtenus pour l'application*

Les comportements obtenus sont résumés Fig. 134. Dans ce tableau sont indiqués : le ou les numéros de cibles correspondant à la Fig.131 qui ont pu bénéficier des

assistances décrites à chaque ligne ainsi que la situation telle que perçue à travers le capteurs d'attributs. Plusieurs mesures peu différentes peuvent obtenir le même jeu d'assistance. Une même cible dans une situation différente après déplacement de l'utilisateur n'a pas les mêmes assistances puisque le contexte a changé.

N°	Situation	Adaptations effectuées
1	 – distance faible – cible visible – ratio à l'écran élevée – pas d'entourage	 Aucune n'assistance n'a eu de confiance assez élevée pour être déclenchée. L'objet n'a pas besoin de valorisation pour être repéré.
2	 – distance plutôt élevée – cible à moitié visible – ratio à l'écran faible – entourage faible (obstacle)	 L'assistance déclenchée est la pulsation visuelle et le bip sonore. La première est ainsi favorisée pour aider à repérer un objet entouré et éloigné, la deuxième pour un objet peu visible.

N°	Situation	Adaptations effectuées
2 ou 6	 – distance plutôt élevée – cible à moitié visible ou visible – ratio à l'écran plus tôt faible – entourage faible (obstacle) ou nul	 Pour la cible N°2, avec quelques variations (de position ou de confiances initiales différentes des attributs et des règles), la représentation de l'espace est ici privilégiée au bip sonore avec l'ajout de la trajectoire (favorisée par la distance élevée ainsi que dans une moindre mesure dans le cas d'un objet caché). C'est également l'assistance du cas N°6 mieux visible et non entouré.
3 ou 4	 – distance élevée – cible non visible – ratio à l'écran faible – entourage élevé ou faible (obstacles)	 L'aide sonore et la représentation de la trajectoire sont déclenchées. La première a une confiance élevée car elle implémente une modalité sonore pour un objet complètement caché, ce qui défavorise également la représentation de l'objet au profit d'une représentation de l'espace.

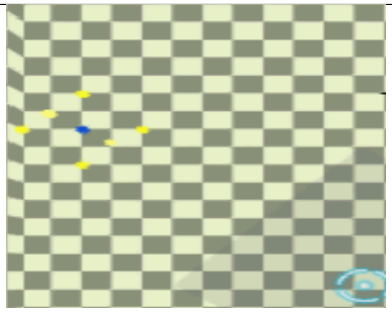
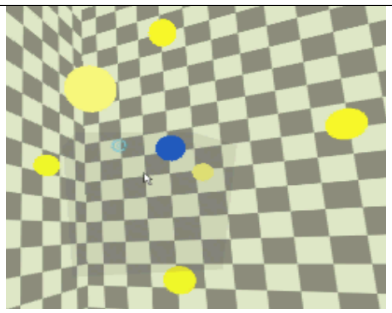
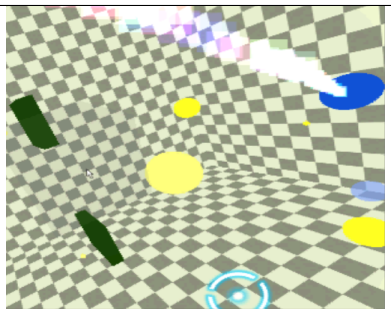
N°	Situation	Adaptations effectuées
5	 – distance plutôt élevée – cible visible – ratio à l'écran faible – entourage élevé (objets)	 L'assistance déclenchée est la pulsation de l'objet cible, permettant de le mettre en valeur visuellement par rapport à l'entourage, ainsi que la représentation de la trajectoire permettant de prendre en compte la distance et de repérer l'objet
5	 – distance faible – cible cachée (derrière la caméra) – ratio à l'écran élevé – entourage élevé (objets)	 Même objet que précédemment mais dans une situation différente : quelques instants après l'illustration de gauche, après avoir dépassé la cible. L'assistance déclenchée est alors l'aide sonore et la représentation de la trajectoire, cohérente avec le contexte actuel.

TABLE 134: Illustration d'assistances variées selon les attributs de l'objet cible

6.2.3 *Conclusion*

Le moteur est sémantique et il est ainsi facile d'ajouter (e.g. la notion de contraires), de restreindre (e.g. le jeu d'outils) ou d'affecter (e.g. imposer un objectif) son raisonnement comme pour cette application. Est illustrée ici sa gestion souple d'un contexte et l'activation d'assistances adaptées à chaque situation. Pour les applications capable de fournir un objectif, l'assistance est plus facile à mettre en place. Ainsi un objet lointain ou caché peut selon sa situation être adéquatement mis en valeur. De plus, ces adaptations évoluent avec le contexte : les valorisations sont dynamiques et ne sont pas fixées par l'identité de la cible mais dépendent de sa situation. En revanche, la richesse globale du raisonnement s'étendant, le temps d'exécution augmente. Il devient trop important pour un usage en ligne avec des cycles de réactions de l'ordre de 20s. Un souci étant que plusieurs cycles sont souvent nécessaires pour obtenir les adaptations adéquates. Par exemple dans le pire des cas précédents, où l'utilisateur ne maitrise pas la navigation et dépasse l'objectif, il faut trois cycles de réactions. Un premier qui récupère les nouveaux attributs, un

deuxième qui désactive les assistances précédentes qui ne sont plus adaptées et un dernier pour obtenir les nouvelles aides grâce à l'impact alors récupéré.

Ainsi deux pistes doivent être explorées par la suite : l'accélération du moteur et son usage hors ligne. Des gains de rapidité sont en effet possibles comme des optimisations des différentes étapes car le moteur n'est encore qu'un prototype. Pour l'exemple lié à cette application, en ajoutant la fonctionnalité de planning, le moteur pourra prévoir qu'une désactivation permet de récupérer une partie de l'impact disponible. Il peut alors appliquer d'ores et déjà une partie des autres réactions pertinentes qu'il avait pu sélectionner. De même, la modification du capteur d'attributs, afin qu'il les fournisse automatiquement après activation (et non plus une fois par demande du moteur) permet de réduire la réflexion d'encore un cycle.

6.3 TEST HORS LIGNE DU MOTEUR

Le temps de réponse du moteur sémantique pour un usage d'assistance à l'interaction 3D est limitant. Néanmoins il est déjà possible de l'utiliser en tant qu'assistance ponctuelle en limitant le nombre de propriétés gérées, notamment les outils ou les jeux de règles utilisés. En revanche, le moteur peut être pleinement utilisé dans le cas d'analyse hors ligne.

6.3.1 *Usage de Matlab et analyse hors-ligne*

Un thread de communication entre Matlab et Virtools a été créé. Ainsi nos environnements virtuels peuvent bénéficier directement de traitements apportés par Matlab. Ce dernier peut également communiquer avec le moteur via Java ou via OSC. Utiliser Matlab comme intermédiaire permet de nombreuses applications :

- simuler une session hors-ligne afin de bénéficier de l'analyse par le moteur. Le lien entre Virtools et Matlab permet de stocker les évènements des capteurs de l'environnement virtuel pour ensuite simuler la session enregistrée hors-ligne. Puis, le lien entre Matlab et le moteur, via Java ou OSC, permet alors d'effectuer une analyse sémantique de l'activité de l'utilisateur pendant cette session simulée.
- effectuer une analyse des résultats sémantiques retournés par le moteur. Avec de nombreuses observations de différentes étapes du raisonnement selon de nombreuses situations, une visualisation spécifique ou un traitement supplémentaire est appréciable, même à partir de données sémantiques compréhensibles séparément.
- obtenir un traitement de données permettant d'extraire une sémantique : permettre un stockage sous Matlab des données permet de bénéficier de traitement de données pour l'environnement virtuel (la création d'un capteur, e.g. la reconnaissance de geste) ou directement pour le moteur (des méta-capteurs potentiels).

En considérant l'analyse hors ligne des résultats sémantiques issus du moteur, il y a deux grand types d'utilisation selon le but recherché :

- L'analyse d'une session enregistrée : le but est de comprendre la progression, l'intention, les difficultés, etc. de l'utilisateur.

– L'aide à la conception : le but est de comprendre les situations ou les informations pertinentes afin d'aider à la conception d'un environnement virtuel, en simulant les situations rencontrées par le futur utilisateur. Cette simulation peut être complètement artificielle ou issue de sessions enregistrées comme précédemment. Elle peut également aider à concevoir le raisonnement même du moteur en illustrant les effets d'entrées spécifiques sur son comportement.

C'est cette deuxième forme que nous allons illustrer par la suite.

6.3.2 *Scénario*

Nous allons illustrer les modifications possibles et l'influence de différents paramètres, de différentes situations sur le raisonnement du moteur. Le cas étudié est le choix de la technique d'interaction. Les règles directement concernées sont résumées Fig. 135 mais l'ensemble du moteur est chargé afin de tenir compte des combinaisons non prévues. Nous partons d'une situation comprenant trois objets et trois techniques de sélection disponibles. La tâche active reste toujours la sélection mais la technique initialement active peut changer. Les descriptions des objets et des techniques sont effectuées Fig.136. Les attributs seront fixes pour les tests, ce qui vis-à-vis des objets sous-entend une même position et vision initiales pour l'utilisateur dans l'environnement virtuel. Les attributs des différentes techniques sont des connaissances a priori. L'objet 1 correspond a priori à un usage de la main virtuelle, l'objet 2 du raycasting et l'objet 3 du 2D picking.

Type	Règles	Sections
Contraires	Gestion générale, notamment pour les attributs et la difficulté	4.5.1.2
Obtention d'objectifs	Un intérêt peut être un objectif	5.3.5
	La tâche courante liée à un objet objectif est un objectif potentiel	5.3.5
	Un sous-objectif est un objectif	4.2.3 4.5.3.2
Jugement de facilité	Selon les objets d'intérêt et la technique courante	5.4.5
	Selon les objectifs et la technique courante	5.4.5
Expression de volontés	Utiliser une technique	5.4.4
	Modifier une technique	5.4.6
	Choisir une technique	5.6.3
Gestion de choix	Lien entre les volontés de choisir et le choix effectué	4.5.4.1

FIGURE 135: Règles ayant un effet notable pour l'analyse hors-ligne

La quantité de description et leur qualité peuvent influer. Des qualifications différentes des techniques et objets ont été ainsi tentées : cela reflète la possibilité d'absence ou de présence d'informations supplémentaires pour certains objets et de la subjectivité des descriptions des techniques. La main virtuelle est considérée comme une technique permettant de gérer correctement les objets proches et bien visibles. Le raycasting est décrit comme gérant bien les objets éloignés, avec

une visibilité angulaire élevée (pour faciliter l'intersection avec le rayon malgré la distance) et avec un entourage plutôt faible (pour éviter la difficulté de sélection entre plusieurs cibles proches). On considère qu'il peut néanmoins mieux gérer les objets peu visibles (avec la possibilité d'orienter le rayon et de sélectionner un objet quelque peu caché). Le 2D picking est décrit comme capable de gérer les objets plutôt éloignés comme plutôt proches et plutôt visibles comme plutôt cachés (avec un pointeur 3D capable de sélectionner un objet complètement aligné et caché par un autre), en précisant chacun de ces cas. Le moteur peut gérer des informations contradictoires même peu cohérentes mais considérons déjà ce premier test et les effets sur le raisonnement.

Attribut	Objet Id1	Objet Id2	Objet Id3	Main virtuelle	Raycasting	2D Picking
Distance	faible (0.8)	élevée (0.8)	élevée (0.5)	faible (0.8)	élevée (0.8)	élevée (0.5) + faible (0.5)
Visibilité	visible (0.8)	visible (0.8)	caché (0.8)	visible (0.8)	visible (0.6)	visible (0.5) + caché (0.5)
Entourage	-	faible (0.8)	-	-	faible (0.6)	-
Visibilité Angulaire	-	élevée (0.8)	-	-	élevée (0.8)	-

FIGURE 136: Paramètres fixes du scénario

Des paramètres d'entrées sont ensuite simulés. Ils correspondent à des faits ajoutés à la base de connaissance. Ils pourraient être obtenus en session enregistrée en observant les ajouts spécifiques de certains capteurs [2]. Ces paramètres sont :

- L'importance accordée à chaque objet : préciser qu'un objet est un intérêt, un objectif ou que sa sélection est un objectif. Ces informations ont toutes été ajoutées pour nos tests avec une confiance de 0.8.
- La technique actuellement active parmi les trois disponibles.
- Des préférences peuvent être ajoutées quant à la volonté d'utiliser ou de choisir une technique en particulier.
- Parmi les autres modifications directes, il est possible de préciser que l'on souhaite modifier la technique ou que la tâche est globalement perçue comme difficile.

Des paramètres sont ensuite mesurés, c'est-à-dire issus d'une analyse par le moteur. Contrairement aux des entrées spécifiques ci-dessus, ils incluent la fusion de l'ensemble des informations par le moteur. Ces paramètres sont :

- la facilité de la tâche ;
- la volonté de modifier de la technique d'interaction ;

2. Obtenus par exemple par une requête effectuée au moteur avec l'identité de la *source* des *faits* ajoutés afin d'éviter la fusion de l'ensemble des informations compatibles.

– la volonté d'utiliser ou de choisir les différentes techniques ;
– le choix de la technique effectué (qui ne serait activée dans la suite du raisonnement que si son niveau de confiance et la gestion de l'impact le permettent)
– la facilité de la sélection d'un objet particulier ;

6.3.3 *Remarques sur la simulation et l'exploitation*

Il est à noter que le nombre de situations simulées (60) ne permet pas une observation complète ni symétrique pour l'ensemble des paramètres (e.g. les importances des différents objets pourraient être séparées en trois paramètres distincts et continus avec leur confiance ; de plus certaines combinaisons ont été peu testées). Les paramètres variables sont représentés en coordonnées parallèles et groupés selon certaines de leurs combinaisons. Cette visualisation permet d'avoir l'ensemble des données représentées simultanément afin de pouvoir comprendre le raisonnement et les relations entre les variables, en soi ou dues à nos mesures. Les mesures manquantes peuvent également être observées par une structure moins élaborée (et non symétrique) sur les première variables. Sur les graphiques sont ainsi représentés :

– l'importance accordée aux objets avec quatre paliers : pas d'information, un intérêt, un objectif et un objectif précis, comme les trois premières variables (Objet1, Objet2 et Objet3) ;
– la technique active (Tech. Act.), avec trois paliers : la main virtuelle, le raycasting puis le 2D picking, ;
– les préférences explicitant la volonté d'utiliser une technique (Pref. Util), avec quatre paliers : pas de précision, une préférence pour la main virtuelle, le raycasting ou le 2D picking ;
– les préférences explicitant la volonté de choisir une technique (Pref. Ch.), avec quatre paliers : pas de précision, une préférence pour la main virtuelle, le raycasting ou le 2D picking ;
– Les options (Opt), peu exploitées encore : le premier palier sans ajout puis le deuxième avec un ajout indiquant que la tâche est difficile et le dernier que l'on souhaite modifier la technique de sélection.
– La mesure de facilité de la sélection (Sel.) dont est représentée la confiance ;
– La mesure de la volonté de modifier la technique de sélection (Modif. Tech.) dont est représentée la confiance ;
– La mesure de la volonté d'utiliser chaque technique de sélection (Ut. Main, Ut. Ray, Ut. Pick) dont est représentée la confiance ;
– La mesure de la volonté de choisir chaque technique de sélection (Ch. Main, Ch. Ray, Ch. Pick) dont est représentée la confiance ;
– La mesure de la valeur du choix de technique de sélection avec trois paliers : la main virtuelle, le raycasting et le 2D picking. Sa valeur de confiance est égale à la valeur correspondante parmi les trois variables précédentes.
– Le calcul sous Matlab du coefficient de variabilité de la volonté de choisir : son écart type divisé par sa moyenne.

Ces calculs peuvent être automatisés. Après simulation de la situation par Matlab selon le script, et l'inscription des faits correspondants dans le moteur, une série de mesures peuvent être effectuées [3] et les réponses du moteur stockées. Le traitement

3. Ou directement un vecteur de mesures obtenu via l'inscription d'un fait de logique du premier ordre.

créant les illustrations et les différents groupes s'automatise également puisque chaque variable mesurée correspond à une information spécifique sémantique.

6.3.4 *Influences sur le choix de la technique d'interaction*

6.3.4.1 *Influence des objets et de leur importance*

Fig. 140 illustre les situations selon l'existence d'une importance (intérêt ou objectifs) accordée aux différents objets. Les précisions de préférences ou d'options ont de fortes influences sur le raisonnement et sont représentées et discutées à part. Fig. 137 illustre l'effet de l'existence d'intérêts, Fig. 138 le cas de l'existence d'objectifs simples et Fig 139 celle des objectifs précis (une situation pouvant répondre à plusieurs critères).

Lorsque des intérêts sont exprimés (Fig. 137) la tâche est considérée comme plus difficile que lorsqu'il y a des objectifs indiqués (Fig.138). Plus la sélection est perçue comme facile, moins la volonté de modifier la technique est grande. Les cas considérés comme les plus faciles sont ceux avec des objectifs précis (Fig. 139). En effet la tâche est plus facile car le moteur accorde plus d'importance aux objectifs. S'ils sont connus avec une bonne confiance, ils lui permettent de calculer avec plus de confiance cette facilité générale. La volonté de modifier la technique en cours diminue alors. En revanche l'usage de chaque technique est connue avec plus de confiance, avec un objectif clair, ainsi la volonté de choisir une technique augmente malgré tout.

L'ordre de grandeur de la variabilité du choix ne change pas drastiquement entre l'existence d'intérêts ou d'objectifs et est globalement réparti selon les situations. Les techniques ne sont donc pas toujours plus faciles à séparer entre elles avec un objectif mais le choix peut être réifié plus facilement car la confiance globale de chaque choix est meilleure.

Différents intérêts et objectifs peuvent être précisés et sont parfois conflictuels. Les objectifs liés aux objets 1 et 2 peuvent provoquer le choix de la main virtuelle ou du raycasting. Néanmoins lorsqu'un objectif précis est spécifié, une tendance apparait malgré les conflits d'attention des différents objets (Fig139). Ainsi un objectif précis associé à l'objet 1 provoque un choix de la main virtuelle avec un bon coefficient de variabilité ou celui du raycasting avec un mauvais coefficient. Réciproquement pour l'objet 2 associé au raycasting avec une bonne variabilité ou à la main virtuelle avec une mauvaise variabilité. L'adéquation entre les objets et les techniques apparait ainsi malgré le reste du contexte.

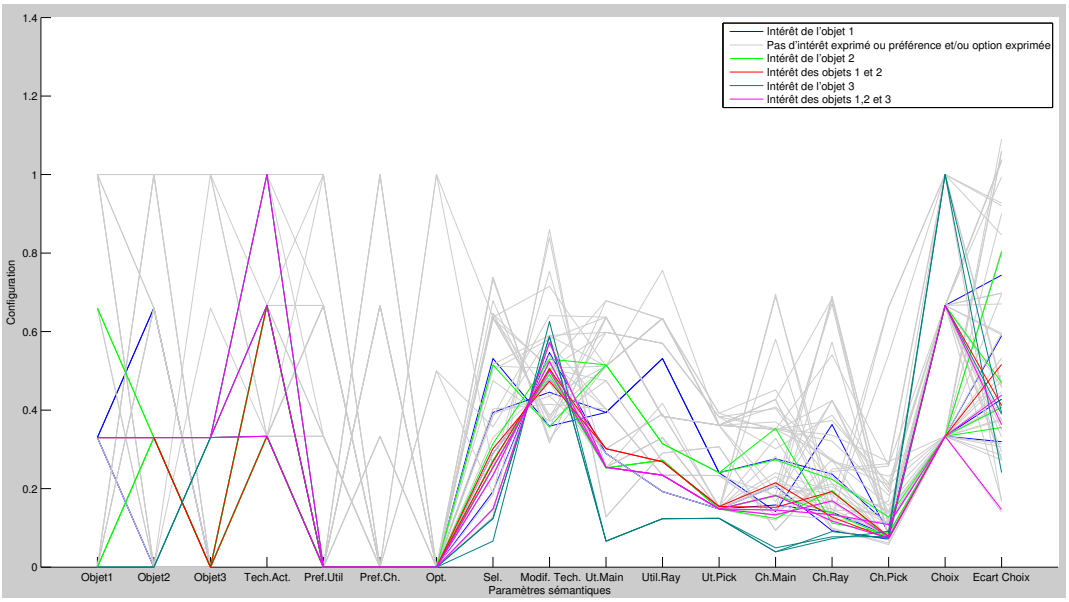


FIGURE 137: Situations groupées selon les connaissances sur les intérêts

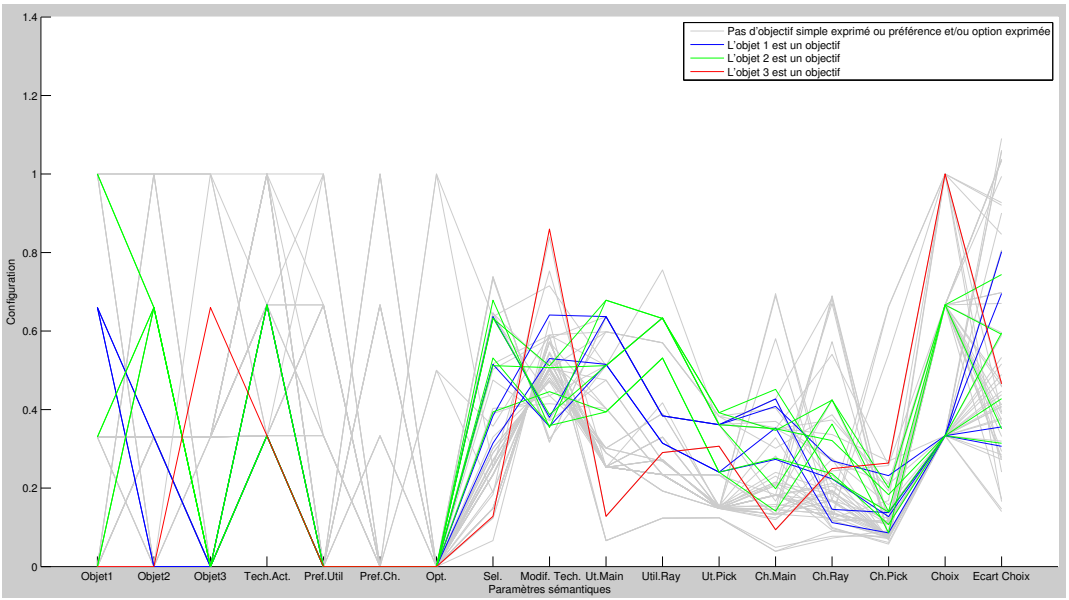


FIGURE 138: Situations groupées selon les connaissances sur les objectifs simples

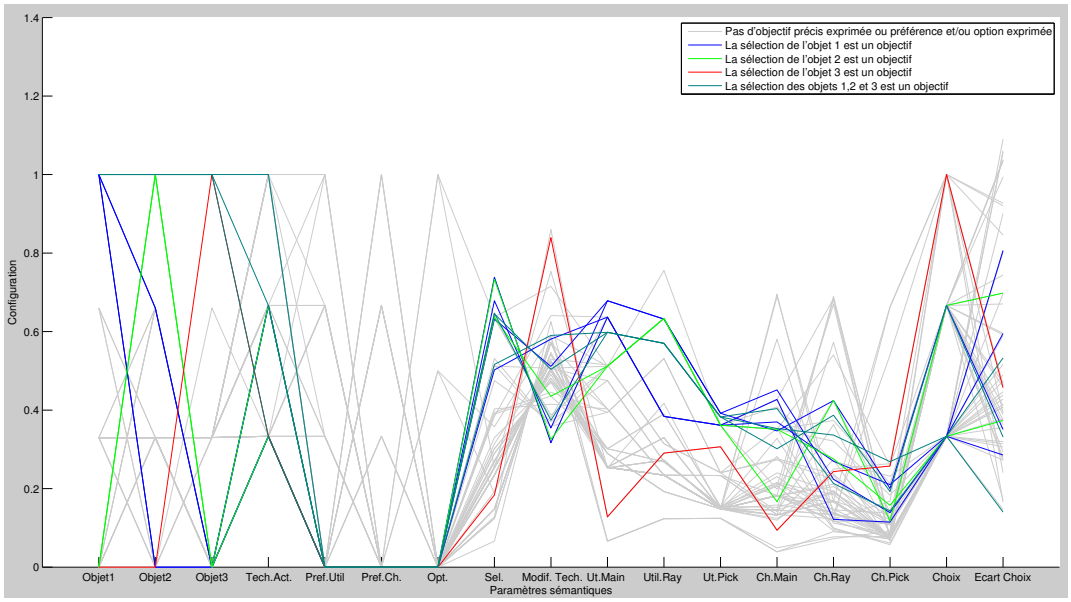


FIGURE 139: Situations groupées selon les connaissances sur les objectifs précis

En visualisant l'ensemble des importances Fig.140, il apparait qu'une importance de l'objet 1 uniquement mène toujours à un choix de la main virtuelle. Une importance de l'objet 2 uniquement mène souvent au raycasting mais parfois à la main virtuelle. Le choix du raycasting est en revanche plus net avec un fort coefficient de variabilité. Une importance de l'objet 3 uniquement peut mener aux différentes techniques mais est le seul chemin menant au choix du 2D picking. Ce sont également les situations qui ont tendance à maximiser la volonté de changer de technique et qui minimisent la confiance de l'ensemble des choix. On pouvait déjà le voir sur les graphes précédents avec le pic isolé de modification lorsque la technique active est la main virtuelle (Fig 137, 138, 139). Une importance partagée entre les trois objets mène le plus souvent à la main virtuelle mais aussi au raycasting (une importance répartie entre 2 et 3 uniquement n'a pas été testée).

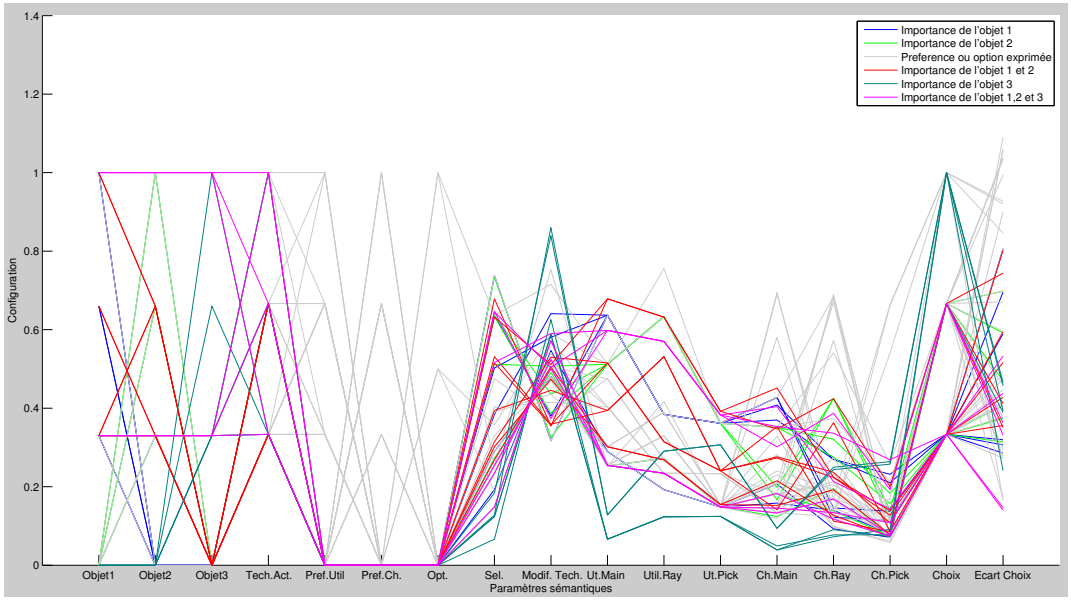


FIGURE 140: Situations groupées selon les connaissances sur les objets

6.3.4.2 *Influence de la technique active*

La Fig.141 est une illustration des situations selon la technique active. C'est la seule illustration qui traite de la facilité de sélection d'un objet en particulier, avec les trois dernières variables supplémentaires. Chacune de ces mesures ne dépend en effet que de la technique active et de l'objet. Sans information additionnelle à ce sujet (ce qui n'a pas été testé), le moteur compare la technique active et les attributs de l'objet (qui ne varient pas au cours de ces tests) pour une requête liée à un objet spécifique. Ainsi la difficulté à sélectionner chaque objet séparément ne dépend que de la technique active à un moment donné.

La technique de main virtuelle est perçue en soi comme la plus adaptée au premier objet et relativement bien adaptée également au second objet. Le Raycasting est en soi adapté à l'objet 2 mais moyennement aux autres. Le 2D Picking est peu adapté en général mais reste le plus adapté, de peu, au troisième objet. Ainsi en ayant peu décrit la technique de main virtuelle, elle est considérée comme potentiellement bien adaptée pour l'ensemble des situations avec un écart de choix toujours élevé lorsqu'initialement active. Le raycasting décrit plus précisément est perçu plus spécifique et associé à l'objet 2. Il possède ainsi deux types de répartition d'écart de choix selon qu'une importance soit donnée à cet objet ou non. Le 2D picking décrit avec des informations contradictoires est perçu comme uniformément peu adapté avec un écart de choix en général faible.

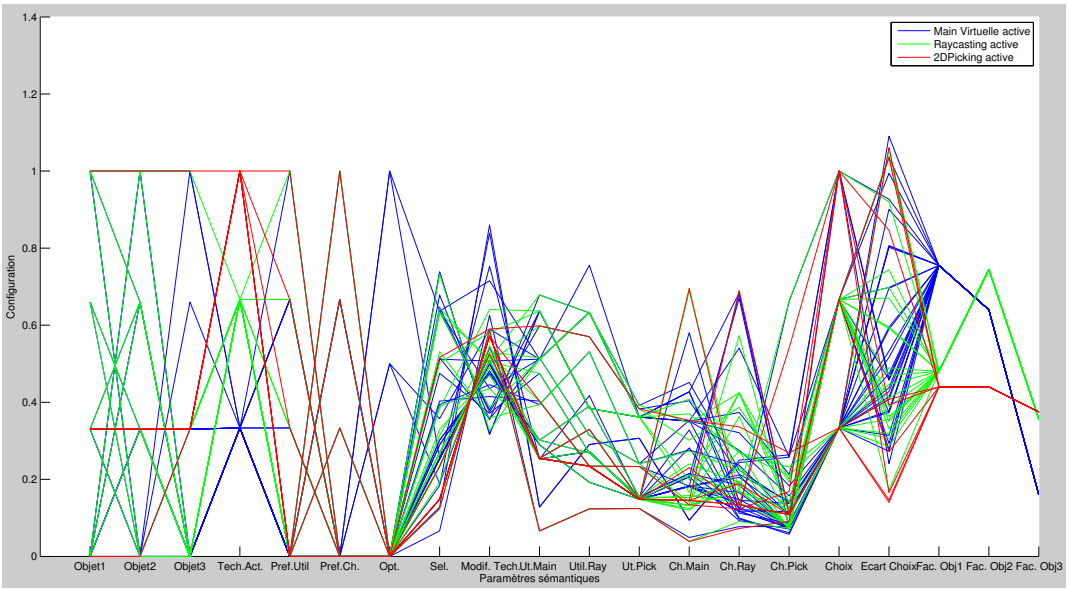


FIGURE 141: Situations groupées selon la technique active

En retirant les situations où une préférence ou une option était exprimée et en illustrant le changement de technique, nous obtenons la Fig.142. On peut voir que les techniques actives sont favorisées par le raisonnement, maximisant leur choix respectif et permettant en général une large marge de variabilité. En revanche lorsqu'un changement de la technique active s'est néanmoins produit, il s'est effectué avec une variabilité du choix faible. Ces changements ont été concordants avec l'apparition d'une volonté de modifier la technique courante élevée. Le changement est donc bien plus souvent provoqué, non pas parce qu'une autre tech-

nique était mieux adaptée mais plutôt parce que la technique courante ne l'était pas.

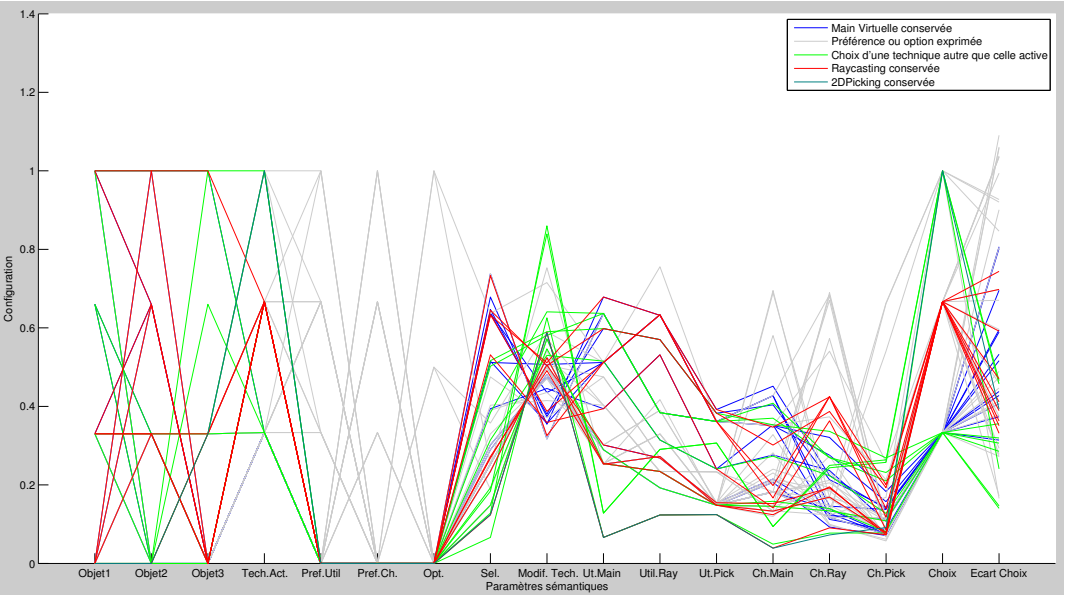


FIGURE 142: Situations groupées selon la technique active sans expression de préférences

Bien qu'il semble que, sans préférence, une technique active n'est jamais abandonnée pour la sélection du raycasting, c'est en fait un artefact dû aux mesures et à leur représentation (Fig.142). Avec le 2D picking actif, le raycasting est parfois choisi (visible notamment sur Fig.145). Mais cela montre que la différence entre les techniques n'est pas perçue suffisamment par le moteur pour choisir le raycasting par rapport à la main virtuelle en l'état.

6.3.4.3 *Influence des préférences sur le choix final*

Le choix peut être influencé par l'ajout d'une volonté d'utiliser ou de choisir une technique. Ces ajouts sont ici qualifiés de préférences, car directement inscrits et non pas issus de règles, comme un a priori de l'utilisateur ou de l'application. Ils modifient la confiance d'étapes intermédiaires du raisonnement et ont donc un poids important sur le choix de la technique, Fig 143. Ainsi aucune volonté de choisir n'a été contredite par le moteur. Bien que possible, une volonté discordante du reste du moteur n'a jamais été suffisante. De plus cette préférence s'ajoute parfois à la volonté interne du moteur selon la situation et permet ainsi les décisions avec les marges les plus élevées du coefficient de variabilité. Les préférences quant a la volonté d'utiliser une technique n'indiquent pas directement une volonté de changement mais le favorisent tout de même. Elles sont également le plus souvent respectées. Elle permettent d'augmenter la marge du choix dans certains cas et font souvent la différence pour entériner un choix dans les cas limites. En revanche cette volonté d'usage n'est pas toujours suffisante notamment lorsque cela provoquerait un changement de la technique active[4], qu'il est plus facile de maintenir ou car la préférence d'usage portait sur la technique de 2D picking, qui est considérée comme peu adéquate par le moteur.

4. Visible par exemple via les transitions entre les variables de technique active et préférences d'usage.

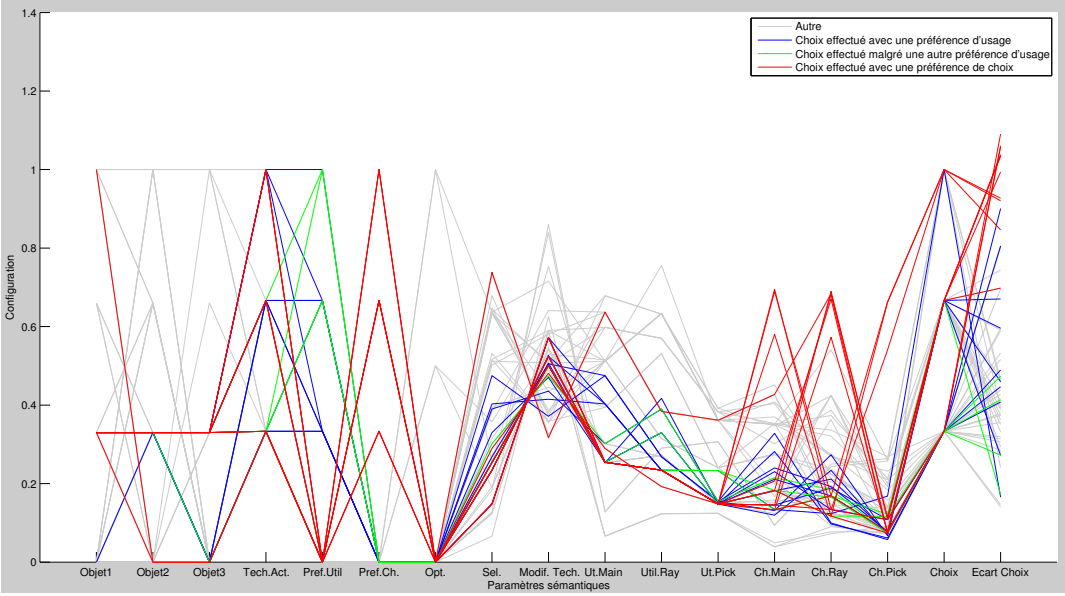


FIGURE 143: Situations groupées selon les préférence et les choix effectués

Fig. 144 illustre les différentes préférences sans distinguer le choix de technique effectué. On peut voir qu'indiquer une volonté de choisir ou d'utiliser maximise ces étapes pour la technique correspondante grâce à la fusion du moteur. Cela ne joue pas sur la volonté de modification qui reste ici toujours modérée. Pour chaque technique, les volontés de choisir ont plus d'influence sur la qualité du choix final, avec une variabilité plus élevée, que la seule volonté d'utiliser une technique.

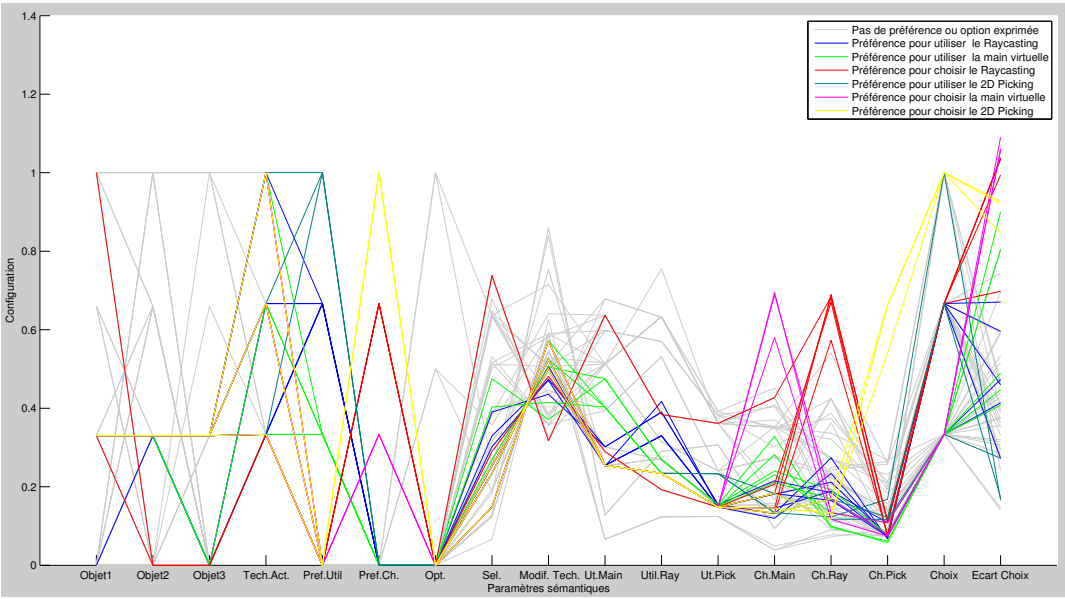


FIGURE 144: Situations groupées selon les préférences exprimées

A noter que le pic isolé rouge de facilité de sélection (Fig. 143 ou 144) est dû à l'inégalité de nos mesures : c'est la seule mesure où un objectif, pour l'objet 1, avait été précisé conjointement à une préférence. La situation était globalement facile notamment car la main virtuelle était active et adaptée (avec une volonté d'usage élevée). Mais avec une volonté de choisir extérieure concernant le raycasting, c'est ce dernier qui a malgré tout été choisi. En revanche la variabilité du choix est

alors bien plus faible par rapport aux autres situations où une telle préférence était indiquée.

6.3.4.4 *Influence de la tâche active et des préférences*

Fig 145 illustre les influences combinées de la tâche active et des préférences d'usages. Les volontés de choisir externes sont prédominantes et ne sont pas représentées. Par contre les volontés d'usages et la technique active semblent être des facteurs d'influence comparable. En effet, une technique active peut être changée pour celle avec préférence d'usage (dans le cas de la main virtuelle ici), mais elle peut être également conservée malgré une autre préférence d'usage (dans le cas du 2D picking et du raycasting ici). Enfin un changement de technique peut se produire pour une autre que celle ayant la préférence d'usage (dans le cas du raycasting ici). Une technique active et possédant une préférence d'usage est généralement conservée.

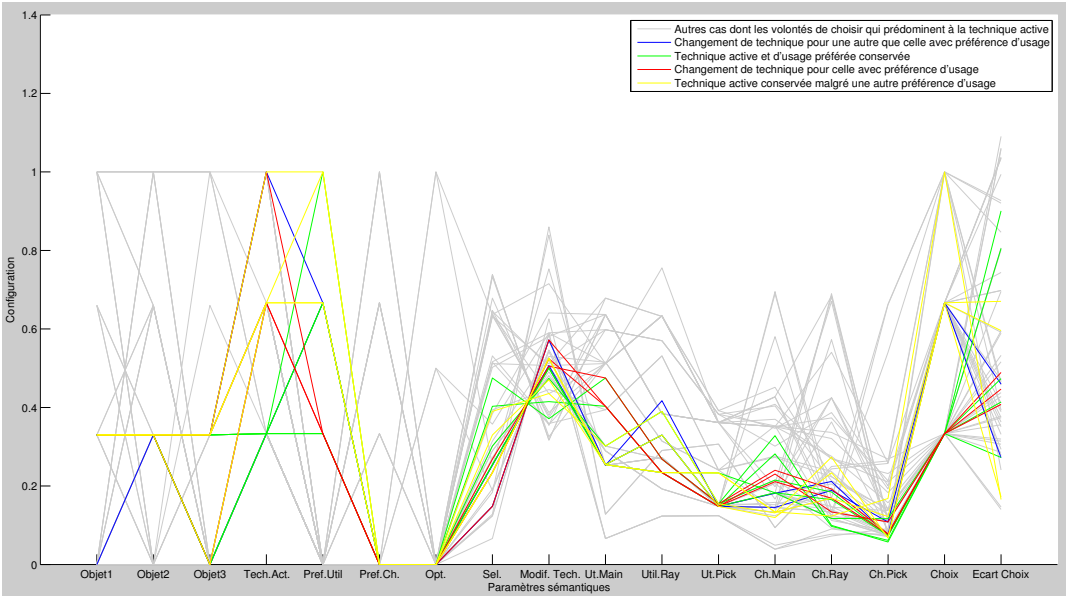


FIGURE 145: Situations groupées selon le choix effectué selon les préférences d'usages et la technique active

6.3.4.5 *Influence d'autres options*

D'autres paramètres ont été peu testés mais ont une influence certaine (Fig. 146). L'ajout d'un fait exprimant la difficulté de l'interaction en général (via un capteur ou une interface utilisateur) ou une volonté de modifier la technique d'interaction (en soi, mais cela peut être conditionnel à un état de repos de l'utilisateur et à la difficulté de la situation) permet de changer le choix effectué par le moteur dans certains cas limites (avec ainsi une faible variabilité). Cela montre à nouveau que chaque étape est modifiable et a son importance. Ici avec la main virtuelle active et une importance donnée à l'objet 2, la difficulté ou la volonté de modification spécifiée favorise le changement de technique pour le raycasting. La difficulté d'interaction a moins d'influence sur le changement de technique, mais pourrait influencer globalement plus de réactions du moteur. La volonté de modifier la technique en cours favorise un changement pour une technique mieux adaptée et contrecarre ainsi l'inertie de la technique active, qui n'est souvent modifiée que lorsqu'elle n'est pas, elle, adaptée.

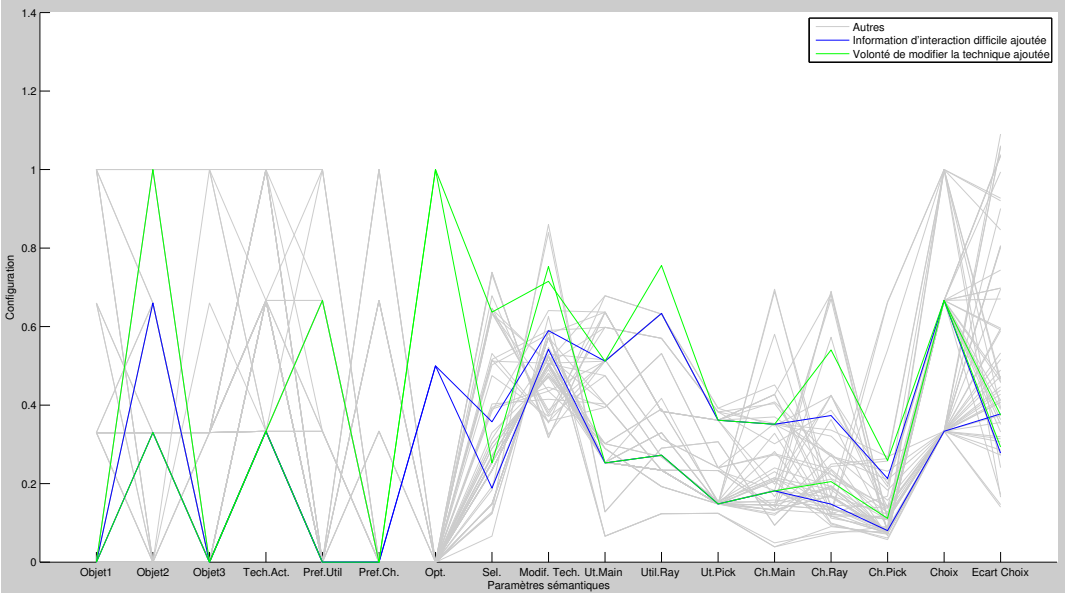


FIGURE 146: Situations groupées selon d'autres informations

6.3.5 *Discussion sur le temps de réponse*

Le moteur avait au préalable été utilisé uniquement pour le processus de réaction générique. Mais il est également possible de poser des questions plus précises. Cela suppose pour l'utilisateur, en ligne, une interface. Pour le concepteur connaissant la grammaire des CGs, les différentes étapes sont directement accessibles. Le moteur peut ainsi être notamment utilisé pour analyser une situation via l'étude des confiances des différentes étapes du raisonnement comme dans cette section.

Ainsi une question plus spécifique n'a pas le même temps de réponse. Formuler une question précise ("l'interaction est-elle anormale ?") plutôt que la demande de réaction (qui correspond à la demande globale "que faire ?") a un temps de réponse de l'ordre de la seconde dans les applications précédentes. Avec l'étude hors ligne, l'algorithme complet, et non plus celui pour les règles aux causes indépendantes est utilisé (Fig. 71 plutôt que Fig. 72, section 4.4.3). Cela nous permet entre autres d'utiliser les règles pour le choix des techniques d'interactions qui justement ont des causes interdépendantes. Par la suite ce processus général devra être optimisé pour être également utilisable en ligne.

Le temps de réponse de l'algorithme général est de l'ordre de 10s à 10 min selon la complexité de la question précise posée. En revanche à l'aide des sauvegardes, tant que la base de connaissance n'a pas été mise à jour, les questions suivantes peuvent être grandement accélérées voire instantanées. Ainsi l'ordre des questions influence dans ce cas le temps de réponse qui est de l'ordre de 20s en moyenne par question. Le processus de réaction complet prend alors de l'ordre de 5 à 40 min pour se terminer (dans le dernier cas, avec l'ensemble des outils et des règles exprimées avec leur complexités finales). Cet algorithme est donc actuellement utilisable uniquement hors ligne.

6.3.6 *Conclusion*

Cette analyse hors ligne a permis d'illustrer de nombreuses influences et comportements de cette partie isolée du raisonnement. Elle permet au concepteur de mieux suivre l'évolution de l'information au travers de règles prévues pour être combinées de nombreuses façons [5]. Ainsi nous pouvons conclure que :

- la description de la technique a un impact. Malgré la différence du nombre d'attributs et de la pertinence des descriptions, les techniques restent comparables et le choix peut être effectué. Une technique décrite avec de nombreux attributs a une spécificité bien perçue et permet une meilleure discrimination du cas idéal mais aussi des autres cas (comme le raycasting). Réduire le nombreux d'attributs risque de rendre la règle très bien adaptée ou pas du tout, selon l'objet (comme la main virtuelle). Des attributs à faible confiance ou contradictoires rendent la technique globalement mal adaptée, même si les adéquations prévues restent vraies (comme entre l'objet 3 et le 2D picking). Elles nécessitent alors une importance plus élevée associée à l'objet pour être perçues (que l'objet 3 soit un objectif).
- les intérêts et objectifs ont l'effet escompté. Il est plus probable qu'une technique soit changée, pour une autre mieux adaptée, lorsqu'un objet est un objectif. En revanche le choix n'est pas pour autant plus facile à faire pour le moteur selon les importances relatives des différents objets. Un effet secondaire est que la situation est globalement perçue comme plus facile par le moteur dès qu'un objectif est présent.
- la technique active possède en pratique une certaine inertie. Le changement de technique est principalement effectuée lorsque la technique active est mal adaptée et non pas lorsqu'il en existe une autre mieux adaptée. Cette propriété est complexe à expliquer et pas directement prévue à ce niveau du raisonnement. Cela vient de la différence de traitement de la technique active pour la volonté de modification et du fait qu'elle soit utilisée comme facteur discriminant dans de nombreuses règles.
- les modifications des étapes intermédiaires ont une forte influence, d'autant plus que l'étape est proche de la conclusion étudiée (comme on peut s'y attendre). Indiquer une interaction difficile a un peu moins d'influence que d'indiquer une volonté de modifier la technique courante, elle même moins influente que la volonté d'utiliser une technique particulière (dont le niveau d'influence est comparable à celle de la tâche active). La plus influente est la volonté ajoutée de choisir une technique particulière qui est toujours respectée dans nos tests.

Ce type d'analyses hors ligne sera par la suite intéressant à mettre en place également pour l'étude de session simulée. Notamment afin de suivre via des mesures spécifiques l'évolution des intérêts ou des objectifs de l'utilisateur, estimés par le moteur, selon les différents capteurs.

5. L'étude visait plusieurs jeux de règles spécifiques mais a été effectuée avec l'ensemble des règles chargées. Une combinaison non prévue et non illustrée par une variable spécifique reste prise en compte dans les résultats.

CONCLUSION ET PERSPECTIVES

PROBLÉMATIQUE

La réalité virtuelle permet la création de mondes spécifiques. Elle vise à concevoir une expérience utilisateur qui soit immersive et interactive. L'interaction 3D avec ce monde est parfois difficile et peut être perçue comme rigide lorsque les techniques ne correspondent pas aux besoins actuels de l'utilisateur. Or comme une seule technique ne permet pas une interaction universelle, il est alors intéressant de concevoir des techniques spécifiques ou des adaptations permettant de répondre à chaque besoin. Mais il est difficile pour le concepteur de pouvoir tenir compte de l'ensemble des situations rencontrées au sein d'une application et entre les applications. De plus il serait intéressant de pouvoir détecter une situation et de proposer à l'utilisateur la technique la plus adaptée, et plus généralement une assistance à son interaction. Ainsi la gestion automatique des assistances et du contexte de l'interaction est bénéfique à l'utilisateur comme au concepteur. L'adaptation et de plus en plus souvent une forme d'adaptativité à la situation sont ainsi des thèmes de recherche pour l'interaction 3D. De nombreux travaux, visant la réalité virtuelle ou non, permettent d'obtenir une telle conscience du contexte. Cependant pour notre approche, nous voulons préserver l'expressivité et l'évolutivité naturellement élevées de la notion de contexte. Une gestion générique de celui-ci nous permet alors d'obtenir des assistances automatiques à l'interaction 3D dans nos applications, sans modification des raisonnements selon leurs possibilités. L'évolution des connaissances peut être faite peu à peu, dynamiquement et à tous niveaux. In fine, l'utilisateur pourra faire évoluer lui-même ces connaissances et ainsi négocier du contexte pertinent à son interaction spécifique.

CONTRIBUTIONS MAJEURES

Les contributions principales de ces travaux sont variées. Premièrement, un processus de gestion générique et sémantique à l'aide de graphes conceptuels permettant une expression compréhensible, riche et ouverte pour les éléments de contexte comme pour le raisonnement, a été implémenté. Ce processus permet un raisonnement sémantique séparé et parallèle aux applications pouvant l'utiliser. Pour cela les applications décrivent leurs outils spécifiques que peut contrôler le moteur et peuvent également fournir à tout moment des informations. L'ensemble est alors combiné puis comparé selon la sémantique des différentes informations. Un degré de vérité, la confiance est introduit au sein du raisonnement ainsi que la gestion du temps, avec la péremption d'évènements et la gestion d'un historique. Ce concept de confiance associé à l'impact des réactions permettent leur comparaison finale malgré des sémantiques différentes. La gestion de l'impact permet également d'éviter de perturber l'utilisateur de ces applications par des changements trop importants.

Ce moteur a ensuite été utilisé pour l'assistance à l'interaction 3D. Pour cela différents outils ont été créés afin d'extraire des informations d'un environnement virtuel ne possédant pas de description sémantique et afin de pouvoir y effectuer des adaptations. Notamment des outils permettant d'obtenir des indices sur l'intention de l'utilisateur sont considérés. Ces différent outils sont ensuite contrôlés à l'aide d'une base de connaissance également créée spécialement. Celle-ci a été pensée afin d'être compréhensible, modifiable et extensive. Ainsi différentes étapes sont introduites dans le raisonnement : l'usage ou l'élaboration de jugement de valeur comme les intérêts et objectifs, les difficultés, les volontés d'action et la présence potentielle d'anomalies mais aussi des jugements de fait comme la disponibilité ou l'état actif. Le moteur permet la gestion du sens accordé aux différents concepts avec par exemple des effets secondaires ou des conditions nécessaires à une activation. La combinaison générale de réflexions sur des éléments isolés d'une même adaptation a été intégrée et permet d'être tant que possible indépendant des capteurs et effecteurs présents. Les réactions automatiquement déclenchées vont de la mise en valeur d'objet au choix de technique d'interaction et à l'accomplissement de la tâche en cours.

Ces raisonnements ont pu être mis en pratique dans des premières applications en ligne liant le moteur et Virtools. La première illustre la compréhension de la situation notamment des différents intérêts et difficultés de l'utilisateur. Avec une base de connaissance certes moins flexible et moins fournie, le processus est alors souvent perçu comme temps réel car transparent et effectué en quelques secondes. Une deuxième application illustre le déclenchement d'assistances adaptées à la situation. La gestion est souple et pertinente mais le processus prend une vingtaine de secondes à s'effectuer et devient trop long pour un usage en ligne. Une dernière application montre une étude hors ligne à l'aide du lien entre le moteur et Matlab. Les influences sur le raisonnement et l'évolution de la compréhension d'un ensemble de situations menant au choix d'une technique d'interaction sont ainsi illustrés. Ce même processus peut être effectué pour analyser des sessions enregistrées.

Le moteur est expressif et versatile. Il peut être utilisé pour une assistance universelle automatique, ponctuelle ou manuelle mais également pour l'analyse d'activité ou la conception hors ligne. L'évolution du temps de réponse favorise un travail futur sur l'optimisation de l'implémentation et un usage actuel du moteur pour l'analyse hors ligne ou l'assistance en ligne n'utilisant qu'un sous-ensemble de ses capacités.

PERSPECTIVES D'IMPLÉMENTATION

Temps de réponses

Des améliorations peuvent être apportées sur le court terme, notamment sur le temps d'exécution. Un des moyens d'accélérer le processus est la sauvegarde pour chaque cycle de calculs, notamment ceux associés à la confiance. Cette sauvegarde est effacée à la fin du cycle ou à la mise à jour de la base de connaissance. Elle a été partiellement effectuée dans la version générale du processus de réaction et a déjà divisé son temps de réaction par trois. Elle n'avait pas été considérée au préalable car elle oblige l'usage d'une interface spécifique pour détecter un changement de

la base de connaissance. De plus ces sauvegardes ne sont pas toujours simples à mettre en place pour conserver la généralité et l'ouverture aux modifications du moteur.

La mise en place de sauvegardes générales nécessite de considérer les variables de la requête. Prolog+CG permet d'appeler des méthodes java notamment celles nécessaires à son propre fonctionnement dont cette fonctionnalité. Il sera alors possible d'effectuer des sauvegardes pour chaque étape y compris pour l'algorithme réservé aux règles avec causes indépendantes. Celui-ci, plus rapide, peut être appelé automatiquement pour les règles respectant ses conditions. Pour cela une première étape pourra être ajoutée et effectuée hors-ligne marquant les règles pouvant être accélérée à l'aide d'un accès à la gestion des contraintes des variables des causes via des méthodes Java. Il est possible que le concepteur les identifie et ajoute le marquage seul, qui n'est qu'une relation sémantique supplémentaire prise en compte par le moteur.

L'analyse sur une base de connaissance fixe permet de construire peu à peu le raisonnement et de l'accélérer en sauvegardant les étapes intermédiaires. Sur le long terme, ce principe peut être utilisé pour obtenir une forme de compilation du raisonnement sémantique avant son utilisation dans une application. Ainsi la base de connaissance dynamique est séparée en cas qui sont utilisés hors ligne pour créer une sauvegarde des réactions adaptées à différentes situations. Le raisonnement sémantique complet peut être ainsi court-circuité dans les cas correspondant et utilisé pleinement autrement. Les modifications dynamiques de raisonnement peuvent toujours être effectuées mais elles ne seront alors pas toujours effectives sans refaire l'analyse hors ligne.

En revanche, tout en conservant les mêmes propriétés, il est intéressant de considérer le passage des traitements ne nécessitant pas directement Prolog à Java, voire de transformer le méta-interpréteur en interpréteur complet (à l'aide des sources disponibles d'Amine) ce qui permet d'accélérer le raisonnement. D'autant plus que les fonctionnalités de l'interpréteur Prolog+CG, via l'appel à ses méthodes Java, sont de plus en plus souvent considérées.

Expressivité

Sur le court terme pour l'expressivité, il serait intéressant de profiter de la gestion des pré-conditions de règles pour obtenir une accélération du raisonnement (en ne débloquant des règles que dans des situations particulières). De nombreuses règles ne sont pas actuellement exprimées de manière optimale et nécessite la correction de quelques bugs rencontrés pour être optimisées. Sur le long terme l'usage des conditions permet d'étendre naturellement le moteur à la gestion de plan (les ressources nécessaires pour une action peuvent être testées en pré-condition, et consommées en post-condition, si les causes sont vérifiées). Cela permettra également une meilleure gestion de la notion de ressources, notamment de l'impact disponible.

Également sur le long terme, une meilleure expression permettra une meilleure interprétation de la confiance et de l'impact. En effet, le degré de vérité est utilisé

comme un degré d'appartenance lors de la gestion des contraires. Ces deux visions différentes peuvent apporter des incohérences. Ainsi il serait intéressant de formaliser et de pouvoir ainsi expliciter l'appartenance séparément. L'expression des différents degrés d'inconnu, l'imprécis, l'incertain et l'incomplet, au sein des graphes conceptuels peut être appréciable également. En effet, la notion d'agrégation est en fait une gestion de complétude, cohérente avec la notion de confiance, permettant le déclenchement des réactions.

De même l'impact est le degré d'effet sur l'utilisateur, par défaut. Mais il est parfois utilisé également comme levier pour signifier qu'une décision est probablement moins pertinente en soi, et que la confiance doit être élevée pour l'appliquer. Il serait intéressant de pouvoir modifier directement la pertinence. Alors, une décision à faible impact mais également souvent peu intéressante pourrait être également moins souvent déclenchée, malgré le fait que le moteur soit rapidement relativement confiant de son usage. Obtenir rapidement le même effet se fait actuellement en augmentant artificiellement l'impact ; ce qui peut être dérangeant ; notamment dans le cas d'une adaptation restant active.

Enfin l'interface utilisateur est une perspective sur le long terme qui permettra, via les ajouts dynamiques dans la base et le raisonnement accessible du moteur, de discuter directement avec le moteur de ce qui est ou non pertinent tant pour les informations que pour les raisonnements. Cette interface pourra être facilitée par la prise en charge des équivalences entre un langage naturel et les graphes conceptuels dans une évolution prochaine de la plate-forme Amine. Elle bénéficiera également beaucoup de l'usage des outils déjà présents dans sa couche ontologique.

PERSPECTIVES GÉNÉRALES : SÉMANTIQUE ET GRAPHES CONCEPTUELS

La volonté de compréhension du contexte implicite n'est pas nouvelle notamment pour l'interaction 3D [8]. L'inclusion de sémantique afin de comprendre l'environnement est de plus en plus envisagée dans tous les domaines. Par exemple, l'évolution du web vers un web sémantique est un projet en cours. Il permet de faciliter la recherche pertinente d'informations parmi l'incommensurabilité des connaissances disponibles sur le web. Basée sur des principes simples, cette inclusion de métadonnées via le langage RDF, puis OWL et ses évolutions n'est somme toute pas si évidente et pose des problèmes de sécurité, de faisabilité et d'évolutivité. Mais le mouvement est en marche. L'extraction de sémantique a été une étape intermédiaire à cet accomplissement afin d'établir des descriptions du contenu déjà présent dans une page HTML. Puis l'inclusion de descriptions supplémentaires a été considérée et est devenue une norme gérée comme les micro-données dans HTML5. Les graphes conceptuels en sont une évolution probable. Une équivalence est possible avec les normes actuelles et ils sont capables d'étendre leur expressivité [78]. Néanmoins l'explosion du temps de calcul de chaque requête est un problème déjà présent dans les normes précédentes.

L'informatique ubiquitaire, les environnements intelligents, l'intelligence ambiante cherchent également à obtenir des assistances pour l'interaction de tout instant et se tournent alors vers l'ontologie et la sémantique. Idéalement, nos gestionnaires d'environnements virtuels devraient à la longue tous proposer une gestion sémantique en plus de la gestion graphique. Néanmoins ce paradigme

n'est pas encore la norme bien qu'activement exploré [46][56][86]. La représentation sémantique d'environnements virtuels est également souvent effectuée à l'aide de réseaux sémantiques, dont les graphes conceptuels sont un type [56][9].

Nos travaux constituent une étape intermédiaire et exploratoire de ces capacités. Comment bénéficier des avantages de la sémantique alors que la plupart de nos applications ne l'intègrent pas ? Un raisonnement parallèle et automatique ainsi que des outils intégrés peuvent être, eux, sémantiques et permettent cette évolution sans poser de contraintes fortes sur nos applications qui restent réalisées de manière classique. Le raisonnement est encore lent mais le choix de l'expressivité via les graphes conceptuels nous permet d'être déjà tournés vers l'avenir car cette méthode semble promise à un usage de plus en plus fréquent pour l'inclusion de sémantique par défaut. Dès lors nos études via l'usage du moteur pourront être réutilisées au sein d'une technologie plus mûre par la suite. L'inclusion de sémantique dans l'environnement peut déjà se faire peu à peu et être contrôlée par le moteur ajoutant aux objets des informations sur leur usages précédents ou sur des réactions conseillées, sans pour autant les déclencher dues aux limitations de temps de réponse. Cela permet une aide à la conception d'environnement virtuel. De plus, l'analyse hors ligne offre un moyen pour la machine de comprendre nos activités au sein d'un environnement et donc au concepteur d'étudier automatiquement des sessions enregistrées. Nos paradigmes de description et de création d'environnements virtuels évolueront au delà de la description graphique afin d'en permettre une description sémantique. De la même manière que l'interaction directe des utilisateurs avec l'environnement perceptible est nécessaire pour la réalité virtuelle, l'interaction directe avec la description sémantique formant le contexte implicite devra être naturellement gérée.

Troisième partie

ANNEXES

"Ou voi zoizo i vole, ou voi pa traka li nana ek lo van"

On voit l'oiseau voler, on ne voit pas ces tracas avec le vent

A

IMPLÉMENTATION DE LOGIQUE DU PREMIER ORDRE

LA PROGRAMMATION LOGIQUE

Une des difficultés du paradigme de programmation de la preuve automatique, appelé aussi raisonnement sémantique, est de maitriser ses combinaisons. C'est d'ailleurs pour cela qu'introduire des traitements automatiques nécessite le contrôle du flux global d'information et donc l'écriture d'un méta-interpréteur. Les combinaisons sont d'autant plus importantes qu'il faut être précis : leur nombre joue ici sur la confiance globale des différents termes. L'écriture du méta-interpréteur ainsi que certaines extensions, introduisant des propriétés particulières globales (le faux, les contraires, le choix, l'agrégation etc.), sont faites en logique du premier ordre (e.g. Fig. 147). Ces écritures sont gérées de telle manière à permettre par la suite au designer de s'exprimer uniquement à l'aide de CGs sans se préoccuper de cette couche logique. Le livre de Bratko [15] permet de programmer de nombreux algorithmes d'intelligence artificielle en Prolog et offre ainsi de très nombreuses extensions possibles au travail déjà effectué.

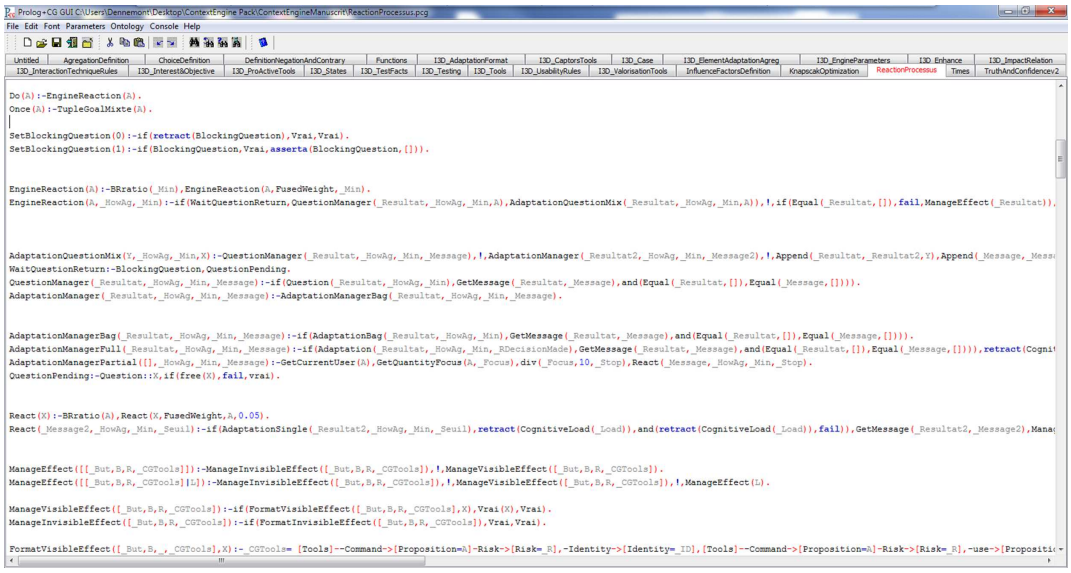


FIGURE 147: Illustration de la logique du premier ordre du moteur

DES RÈGLES CLASSIQUES ET DES RÈGLES PROCESSUS

Les règles logiques sont en pratique séparées en deux grand types. Le premier est celui des règles naturellement compatibles avec le chainage arrière, qui définissent une information dont des causes doivent être vérifiées au moment où l'information est nécessaire au raisonnement. C'est une règle classique, qui s'écrit facilement sous notre format spécifique, reliant des CGs. Cependant d'autres règles implémentent en fait un processus : les effets ne sont en pratique que les descriptions d'un processus et de ses arguments tandis que ses causes représentent les différentes étapes de traitement. Celles-ci sont souvent plus faciles à écrire directement en logique du premier ordre.

Ce type de processus peut être relié à l'expression d'un CG. Par exemple, une règle définie la structure $[X] - type -> [Y]$, via un processus qui vérifie que X et Y soient des concepts non vides avant de tester la relation de parenté au sein de l'ontologie. Cette règle est indépendante et représente une des manières d'interpréter l'expression sans empiéter sur les autres descriptions (par exemple, la présence de cette relation entre deux concepts non parents dans la base, l'existence de *type* de *choix* par exemple).

Ces processus doivent être maitrisés. En effet, lorsque des unifications précédentes ne permettent pas la résolution des causes suivantes, l'interpréteur Prolog effectue un retour arrière. Celui-ci tente de trouver un autre chemin avec d'autres évaluations des termes rencontrés, en revenant en peu à peu en arrière dans les expressions (Fig. 148). Un retour arrière est effectué à partir du terme dont l'unification a échoué, ou à partir du dernier terme pour des solutions supplémentaires. Les termes rencontrés pendant ce chainage arrière sont à nouveau évalués. Leurs éventuels sous-termes sont alors eux-même évalués dans l'ordre de lecture classique. Dans le cas où ces causes étaient des étapes d'un processus, revenir en arrière n'est pas toujours souhaité ni même défini. Ces retour arrières peuvent également être provoqués par des demandes de solutions supplémentaires (notamment via l'usage de l'opérateur "findall" permettant d'obtenir l'ensemble des solutions). Le retour arrière peut être empêché par l'opérateur "cut" (noté !).

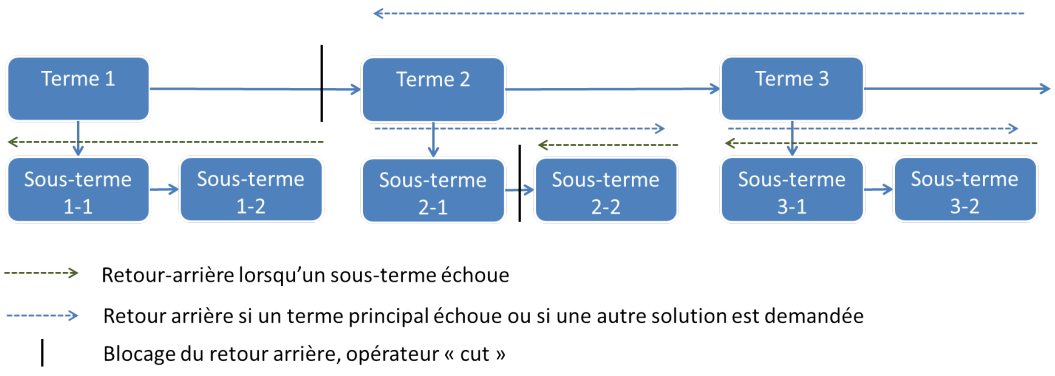


FIGURE 148: Illustration du principe de retour arrière

Ainsi, il faut prévoir la possibilité de chainage arrière pour chaque règle. Celui-ci non maitrisé peut ralentir le raisonnement, provoquer des solutions supplémentaires non désirées voire provoquer des erreurs (une perte de répétitivité, une dépendance

de l'ordre des informations, une interruption du raisonnement et, dans le pire des cas, une boucle infinie). Pour obtenir des règles indépendantes il faut alors pouvoir obtenir l'ensemble des solutions désirées en ne permettant le retour arrière que lorsque nécessaire.

EXEMPLE DE LA GESTION DES CONTRAIRES

Par exemple, pour la gestion des contraires, plusieurs règles de type processus ont été définies. Il faut ainsi pouvoir gérer la véracité et la confiance d'une demande spécifique exprimant un attribut dont le contraire est présent dans la base. Mais il faut également pouvoir trouver la liste des attributs à partir d'une structure vide incluant les contraires déductibles de ceux présents dans la base. Cette règle est illustrée Fig. 149. Elle introduit des verrous logiques et des blocages de retour arrière. Son expression statut sur une structure d'attribut vide afin d'être utilisée automatiquement par le chainage arrière classique au moment où cette information est nécessaire. Cette même structure est évaluée au sein du processus lié à la règle afin de trouver les attributs explicitement décrits dans la base de connaissance. Un des chemins de cette évaluation passe donc par cette règle, l'ayant elle même appelée, et doit ainsi être bloqué pour éviter la boucle infinie (rôle des premiers tests et supposition du verrou). Après qu'un attribut soit trouvé (par un autre moyen donc que cette règle) et traité pour obtenir un contraire, ou après l'échec de cette recherche, le verrou est retiré. Cependant, un retour arrière peut être demandé. Il doit alors permettre de trouver l'ensemble des contraires de chaque attribut explicitement décrit. Ainsi le retour arrière est permis mais limité au dernier bloc. Ce dernier commence par tester la présence du verrou, qui n'est plus dans la base pour ce retour arrière et doit alors être à nouveau affirmé pour éviter les boucles infinies. Il est donc possible d'effectuer des évaluations multiples ou simples, sans effets secondaires indésirables et en conservant le sens attendu pour chaque possibilité.

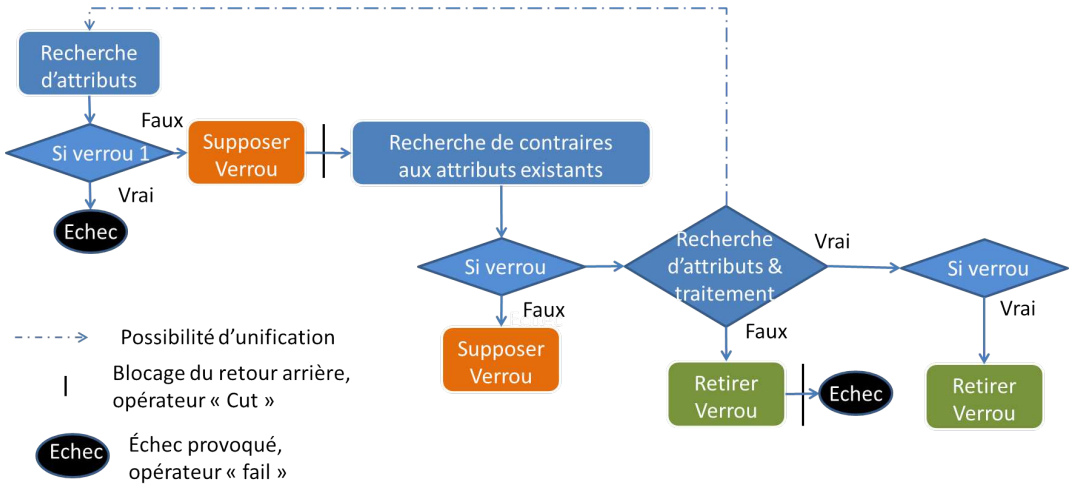


FIGURE 149: Conception d'une règle permettant l'obtention de contraires

En pratique l'ajout et la suppression de verrou changent la base de connaissance et constituent en soi des chemins supplémentaires. Il faut complexifier la structure avec un deuxième verrou pour s'assurer qu'un "findall", recherchant l'ensemble des chemins ne puissent obtenir des chemins à l'infini (Fig. 150). Le verrou 2 ajouté permet de distinguer le cas du retour arrière. Lorsqu'un retour arrière a

épuisé toutes les possibilités et que la recherche interne d'attributs et son traitement échouent, le verrou 2 reste présent. En faisant échouer la première tentative de réutilisation de la règle à l'issue de ce dernier cas, il permet un fonctionnement adéquat du findall.

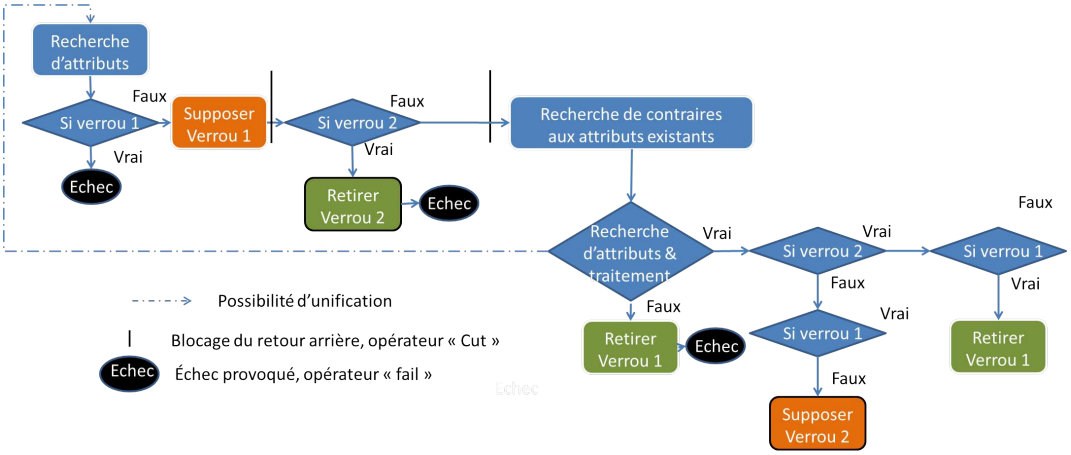


FIGURE 150: Conception implémentée de la règle permettant l'obtention de contraires

GESTION DES DIFFÉRENTES ÉCRITURES

Pour permettre à la fois des libertés d'écritures de CGs et leurs interprétations correctes par le moteur, la tâche du concepteur voulant réutiliser facilement ses règles et/ou les rendre tant que possible interopérables est d'autant plus complexe. La gestion des différentes interprétations est un casse-tête sans fin : par exemple, il est a priori possible écrire un *fait* comme description d'un autre *fait* et cela devrait pouvoir être interprété. L'écriture naturelle a été ainsi un souci constant pendant la conception du moteur. Bien que cohérente avec notre volonté d'accès et de modification facile, elle est consommatrice en ressources et constitue un travail de recherche en soi. Imposer une interface pour les modifications et les ajouts du moteurs permettra ainsi de décrire des règles traitant les entrées utilisateur sans pour autant qu'elles soient évaluées à chaque étape du raisonnement. Des nombreuses interprétations actuellement intégrées, la plupart devraient être ainsi dédiées à cette interface afin de ne pas perdre en expressivité tout en permettant une exécution plus efficace.

A noter que des différences existent entre les versions illustrées dans le manuscrit et les versions implémentées. Notamment l'ajout d'étiquette (la version spécifique à Amine des co-références en plus rapide) n'est pas toujours bien interprété et donc souvent remplacé par une variable avec un test de filiation ontologique supplémentaire. De nombreux tests sont ainsi effectués sans raison fondamental et le moteur gagnera à leur correction future. En dernière remarque, un terme de logique du premier ordre X, souvent représentant un processus, peut être intégré au besoin au sein d'un CG via la structure $[\mathrm{Proposition} = \mathrm{X}] - \mathrm{estUn} - > [\mathrm{TermePCG}]$. Ce CG est vrai si X est un terme de logique du premier ordre vrai. Cela permet de ne pas limiter l'expression dans le format de règles du moteur (a priori en tant que condition) tout en bénéficiant des propriétés introduites pour les causes (transfert de confiance etc.).

B

IMPLÉMENTATION DE LA RECONNAISSANCE DE GESTES

DÉFINITION DES SYMBOLES ÉMIS

Utilisant l'algorithme présenté section 5.3.4.2, il faut, à partir des capteurs disponibles, construire ce que seront les symboles émis. Une première manière serait de considérer comme observations directement les données capteurs. Néanmoins il faudrait alors pouvoir gérer quantité de symboles différents, pas tous appropriés. Coder notre information, notamment via une quantification, a plusieurs avantages (d'où la compétitivité des modèles de Markov discrets). Il permet d'ajuster la pertinence de chaque symbole et ainsi améliorer dès cette étape la robustesse à la variation ou au bruit. On ne fait également pas d'hypothèse sur la densité de probabilité sous-jacente aux données.

Le codage dépend de la grandeur caractéristique du mouvement que l'on utilise. Nos capteurs nous fournissent la position et/ou l'orientation. Dans les deux cas, il est avantageux de travailler sur la dérivée de ces grandeurs. Ainsi, cela permet :

- De suivre la définition du geste communicatif qui est le mouvement porteur de sens.
- De s'affranchir du repère.
- De ne pas avoir à déterminer une partition de l'espace adaptée.

C'est pourquoi à partir de la position, on calcule le vecteur vitesse instantanée. Puis on code ce vecteur selon son orientation, c'est-à-dire en considérant uniquement la direction du geste (figure 151). La norme du vecteur n'est pas utilisée dans la définition des symboles car on ne désire pas que la rapidité d'exécution soit un critère pertinent. Ainsi le codage dans un plan [**?**] est le suivant :

$$Symbole_{xy} = \frac{\lfloor \arctan(\frac{Vitesse_y}{Vitesse_x}) \rfloor}{\frac{2 * \Pi}{N_{xy}}}$$

où $Vitesse_x$ est la composante suivant x de la vitesse, $Vitesse_y$ est la composante suivant y de la vitesse, N_{xy} est le nombre de directions considérées dans le plan $[xy]$ (figure 151). Cette formulation permet donc de réduire l'information du vecteur vitesse de deux dimensions à un seul symbole correspondant à sa direction.

Enfin on transpose ce codage en trois dimensions, en l'utilisant une fois dans chaque plan. Donc avec N_{yz}, dans le plan $[yz]$, on code $Symbole_{yz}$. N_{yz} et N_{xy} sont choisis égaux dans la suite et on évoquera ainsi indistinctement le nombre de directions dans un plan. On réunit ces deux informations en un seul symbole :

$$Symbole = (Symbole_{yz} - 1) \times N_{xy} + Symbole_{xy}$$

Un symbole correspondant à l'état de repos, lorsque le mouvement est théoriquement nul, a été rajouté ce qui définit un alphabet de $N_{xy} \times N_{yz} + 1$ observations.

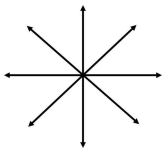


FIGURE 151: Considered plan [xy] direction orientations for $N_{xy} = 8$

En changeant la fonction de codage des familles de gestes peuvent être reconnues à l'issue de l'apprentissage d'un geste : coder les écarts à la direction moyenne pour reconnaitre l'ensemble des gestes rectilignes ou l'évolution du vecteur perpendiculaire au plan pour des gestes entourant une cible.

GESTION DU SEUIL

La première méthode de classement consiste à considérer le geste le plus probable puis de vérifier que cette vraisemblance est supérieure à un seuil constant pour valider la détection. Cette méthode ne permet de bons résultats que dans des cas très particuliers et il convient de la modifier. Il est en pratique quasiment impossible d'utiliser un seuil constant pour départager les gestes. En effet ceux-ci peuvent être de longueurs différentes, or plus un geste est long, moins il est probable. Nous n'allons donc pas comparer directement la probabilité des gestes mais leur écart par rapport à un seuil variable, décroissant en fonction du nombre de symboles du geste. Un premier seuil utilisable est celui où les probabilités de transition et d'émission seraient uniformes. Celui-ci s'écrit donc pour un geste comprenant L symboles :

$$Seuil\,uni(L) = (\frac{1}{Nombre\,total\,de\,symboles\,permis})^{L}$$

Ce seuil utilisé seul a de très mauvais résultats lorsque l'on considère un état de repos. En effet celui-ci est beaucoup plus probable que les autres symboles pour la vraisemblance associée au seuil. Un autre choix pertinent de seuil décroissant est celui correspondant à une probabilité gérée elle aussi par un HMM. Nous pouvons de plus l'entrainer à reconnaitre le bruit et les gestes hors alphabet. Une combinaison de ces deux solutions permet de meilleurs résultats. Une constante est rajoutée à cela afin de régler plus finement la détection. Son réglage permet souvent de doubler le taux de réussite. Le seuil aura comme forme générale, avec une pondération P :

$$Seuil(L) = P \times Seuil\,uni(L) + (1 - P) \times Seuil\,HMM(L) + Gain$$

Nous pouvons rendre automatique le réglage grossier de cette constante *Gain*. Il suffit de relever le maximum des scores (probabilité du geste moins celle du seuil) de chaque geste ($Score\,max$). Pour être détectable ces scores doivent être positifs. Nous allons assurer cela en donnant au gain une valeur proportionnelle au plus petit de ces maximums. Ainsi si ce dernier est positif, on donne au gain la valeur $0.9 * min(Score\,max)$, ce qui permet au moins probable des gestes de rester détectable tout en réduisant d'éventuelles fausses détections. S'il était négatif on règle cette

valeur à $1.1 * min(Score\,max)$, ce qui permet au moins probable des gestes de devenir détectable. Avec cette automatisation, la détection est rendue beaucoup plus robuste par rapport à la combinaison de gestes que l'on veut détecter, et finalement nous n'avons plus besoin (pour l'instant) du seuil uniforme dans la très grande majorité des cas. Finalement :

$$Seuil(\mathrm{L}) = Seuil\,HMM(\mathrm{L}) + Gain\,auto$$

Malheureusement certains gestes risquent d'être incompatibles. En effet le principe de la détection est de considérer la probabilité d'une séquence de symboles. Or si un geste plus long contient intégralement un autre geste plus court que l'on souhaite aussi détecter (par exemple le geste "Z" contient entièrement le geste "Droite"), le plus long ne sera jamais repérable. Il faut alors déclarer cette possibilité de conflit, dépendant du choix de l'alphabet. Le programme introduit alors un délai (selon leur différence de longueur) dans la détection du geste le plus court afin de repérer si celui-ci faisait partie d'un plus grand ensemble ou non.

RÉSULTATS DE LEUR ÉVALUATION

Le module a été évalué [34] à partir d'un alphabet contenant 4 gestes simples, les quatre directions cardinales et deux gestes complexes, le geste "z" et le geste de "Boucle gauche". Les gestes simples ont eu des réussites toutes supérieures à 80% dans le cas de test le plus adapté. Les gestes complexes ont des performances moindres, inférieures de l'ordre de 20%. Le choix de l'alphabet de geste est fondamental. Il doit assurer de bonnes performances, par exemple en évitant les gestes à conflit voir les gestes antagonistes, sources d'erreurs supplémentaires. Il faut néanmoins privilégier des métaphores et des gestes d'actions intuitifs, car c'est là tout l'intérêt du module. De plus l'évaluation confirme cela, avec de meilleurs résultats pour le geste complexe qui a le désavantage d'être en conflit mais dont le symbole est connu de tous. Les gestes complexes sont en général à réserver à des situations peu courantes ou nécessitant un très faible taux de fausses reconnaissances.

RECONNAISSANCE DE GESTES ET CONTEXTE

Une détection plus poussée peut être envisagée. Ainsi on pourrait exploiter les gestes partiellement reconnus, via uniquement une partie du geste (avec plusieurs fenêtres d'observations) ou alors ceux avec un score élevé mais n'ayant pas dépassé le seuil. Cela servirait de période de transition , permettant d'approfondir la connaissance réciproque entre l'utilisateur et le module, comme pour le contexte. C'est un peu ce que l'on peut observer dans le cas de la "Boucle gauche" dont les résultats s'améliorent doucement dans notre évaluation, malgré le durcissement de l'application de reconnaissance : l'utilisateur apprend peu à peu le geste. Il est également envisageable de gérer une modification de la sensibilité ou de la reconnaissance de ces gestes par le moteur selon le contexte. Également à l'aide des variations du geste perçues on pourrait relever des nuances de l'intention. Cela correspondrait à un second ordre dans la reconnaissance et à un capteur supplémentaire d'intention pour le moteur.

BIBLIOGRAPHIE

[1] Ruth Aylett and Michael Luck. Applying artificial intelligence to virtual reality : Intelligent virtual environments. *Applied Artificial Intelligence*, 14(1) : 3–32, 2000. ISSN 0883-9514. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.26.4765&rep=rep1&type=pdf.

[2] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4) :263–277, 2007.

[3] Avron Barr, Edward A Feigenbaum, and Paul R Cohen. *The handbook of artificial intelligence*, volume 1. Pitman, 1981.

[4] Mary Bazire and Patrick Brézillon. Understanding Context Before Using It. *Context*, pages 29 – 40, 2005.

[5] Alain Berthoz and Jean-Louis Vercher. *Le traité de la réalité virtuelle : Volume 1, L'homme et l'environnement virtuel*, volume 1. Presses de l'Ecole des Mines, 2006.

[6] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. *Pervasive and Mobile Computing*, 2010. ISSN 15741192. doi : 10.1016/j.pmcj.2009.06.002.

[7] Antonis Bikakis, Theodore Patkos, Grigoris Antoniou, and D. A survey of semantics-based approaches for context reasoning in ambient intelligence. *Ambient Intelligence*, 2008. URL http://www.springerlink.com/index/KN57JT03K63N345L.pdf.

[8] Richard A. Bolt. Put-that-there : Voice and gesture at the graphics interface. *of the 7th annual conference on Computer graphics*, pages 262–270, 1980. URL http://portal.acm.org/citation.cfm?id=800250.807503.

[9] B. Bonis, J. Stamos, S. Vosinakis, I. Andreou, and T. Panayiotopoulos. A platform for virtual museums with personalized content. *Multimedia Tools and Applications*, 42(2) :139–159, October 2008. ISSN 1380-7501. doi : 10.1007/s11042-008-0231-2. URL http://www.springerlink.com/index/10.1007/s11042-008-0231-2.

[10] Pierre Boudoin. *L'interaction 3D adaptative : une approche basée sur les méthodes de traitement de données multi-capteurs*. PhD thesis, Université d'Evry, 2010.

[11] Pierre Boudoin, Samir Otmane, and Malik Mallem. Fly Over , a 3D Interaction Technique for Navigation in Virtual Environments Independent from Tracking Devices. *Virtual Reality*, (Vric), 2008.

[12] G Bouyer, P Bourdot, and M Ammi. Supervision of Task-Oriented Multimodal Rendering for VR Applications. *Communications*, 2007.

[13] Doug A Bowman, Ernst Kruijff, Joseph J LaViola Jr, and Ivan Poupyrev. *3D user interfaces : theory and practice*. Addison-Wesley Professional, 2004.

[14] Doug A Bowman, Jian Chen, Chadwick A Wingrave, John Lucas, Andrew Ray, Nicholas F Polys, Qing Li, Yonca Haciahmetoglu, Ji-sun Kim, Seonho Kim, Robert Boehringer, and Tao Ni. New Directions in 3D User Interfaces. *International Journal*, 5, 2006.

[15] Ivan Bratko. *Prolog Programming for Artificial Intelligence*. Number 4th. International Computer Science Series, Aout 2011.

[16] Patrick Brézillon. Context in Problem Solving : A Survey. *Artificial Intelligence*, pages 1–40, 1995.

[17] Patrick Brézillon. Modeling and Using Context : Past , Present and Future Abstract :. *Knowledge Acquisition*, pages 1–58, 1999.

[18] Patrick Brézillon. From expert systems to context-based intelligent assistant systems : a testimony. *Engineering*, 26 :19–24, 2011. doi : 10.1017/S0269888910000366.

[19] Patrick Brézillon, Laurent Pasquier, and Jean-charles Pomerol. Reasoning with contextual graphs. *European Journal Of Operational Research*, 136 :290–298, 2002.

[20] Jeffrey Cashion, Chadwick Wingrave, and Joseph J Laviola Jr. Dense and Dynamic 3D Selection for Game-Based Virtual Environments. 18(4) :634–642, 2012.

[21] Augusto Celentano and Michele Nodari. Adaptive interaction in Web3D virtual worlds. *Proceedings of the ninth international conference on 3D Web technology - Web3D '04*, 1(212) :41, 2004. doi : 10.1145/985040.985047. URL http://portal.acm.org/citation.cfm?doid=985040.985047.

[22] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. What are ontologies, and why do we need them ? *IEEE Intelligent Systems*, 14(1) :20–26, January 1999. ISSN 1094-7167. doi : 10.1109/5254.747902. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=747902.

[23] M. Chein and ML. Mugnier. *Graph-bases Knowledge Representation : Computational Foundations of Conceptual Graphs*. Springer, 2009.

[24] M. Chein, M L Mugnier, and G Simonet. Nested Graphs : A Graph-based Knowledge Representation Model with FOL Semantics. *Elements*, 1998.

[25] Guanling Chen, David Kotz, et al. A survey of context-aware mobile computing research. Technical report.

[26] Luca Chittaro and Roberto Ranon. Adaptive 3D Web Sites. *Environments*, 2007.

[27] Ali Choumane, Gery Casiez, and Laurent Grisoni. Buttonless clicking : Intuitive select and pick-release through gesture analysis. *2010 IEEE Virtual Reality Conference (VR)*, pages 67–70, March 2010. doi : 10.1109/VR.2010.5444810. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5444810.

[28] Paolo Coppola, Vincenzo Della Mea, Luca Di Gaspero, Raffaella Lomuscio, Danny Mischis, Stefano Mizzaro, Elena Nazzi, Ivan Scagnetto, and Luca Vassena. *AI Techniques in a Context-Aware Ubiquitous Environment*, pages 150–180. 2009.

[29] Henriette Cramer. *Peoples's responses to autonomous and adaptive systems*. PhD thesis, 2010.

[30] Madalina Croitoru, Alain Gutierrez, and Marie-laure Mugnier. Translations between RDF (S) and Conceptual Graphs. In *Conceptual Structures : From Information to Intelligence 18th International Conference on Conceptual Structures (ICCS 2010)*, number Iccs, 2010.

[31] Waltenegus Dargie and Thomas Springer. Integrating facts and beliefs to model and reason about context. In *Distributed Applications and Interoperable Systems*, pages 17–31. Springer, 2007.

[32] Harry S Delugach. Charger : A graphical conceptual graph editor. In *Workshop on Conceptual Graphs Tools at the 9th International Conference on Conceptual Structures*, 2001.

[33] Alexandre Demeure and G. Calvary. Le modèle d'évolution en plasiticité des interfaces : apport des graphes conceptuels. *Actes d'IHM*, 2003. URL http://iihm.imag.fr/publs/2003/ihm2003.evolution.pdf.

[34] Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. A Discrete Hidden Markov Models Recognition Module for Temporal Series : Application to Real-Time 3D Hand Gestures. In *Proc. of the 3rd IEEE International Conference on Image Processing Theory, Tools and Applications (IPTA 2012)*, pages 299–304, Istanbul, Turquie, 2012. URL http://hal.archives-ouvertes.fr/hal-00749097.

[35] Yannick Dennemont, Guillaume Bouyer, Samir Otmane, and Malik Mallem. 3D interaction assistance in virtual reality : a semantic reasoning engine for context-awareness : from interests and objectives detection to adaptations. In *Proc. of the 8th International Conference on Computer Graphics Theory and Applications (GRAPP 2013)*, pages 349–358, Barcelone, Espagne, 2013. URL http://hal.archives-ouvertes.fr/hal-00761912.

[36] A.K. Dey and G.D. Abowd. Towards a better understanding of context and context-awareness. In *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*, volume 4, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.4833&rep=rep1&type=pdf.

[37] Paul Dourish. What we talk about when we talk about context. *Personal and ubiquitous computing*, 8(1) :19–30, 2004.

[38] Paul Dourish. *Where the action is : the foundations of embodied interaction*. The MIT Press, 2004.

[39] E.Knutov, P. De Bra, and M. Pecheniskiy. *AH -12 years later : a ceomprehensive survey of adaptive hypermedia methods and techniques*, pages 5–38. 2009.

[40] Scott Frees. Context-driven interaction in immersive virtual environments. *Virtual reality*, 14(4) :277–290, 2010.

[41] Loïc Fricoteaux, Indira Thouvenin, Jérôme Olive, and Paul George. Evidential network with conditional belief functions for an adaptive training in informed virtual environment. In *Belief Functions : Theory and Applications*, pages 417–424. Springer, 2012.

[42] David Genest and Eric Salvat. A platform allowing typed nested graphs : How cogito became cogitant. In *Conceptual Structures : Theory, Tools and Applications*, pages 154–161. Springer, 1998.

[43] James J Gibson. The theory of affordances. *Perceiving, acting, and knowing : toward an ecological psychology*, 1977.

[44] T Gruber. Toward principles for the design of ontologies used for knowledge sharing ? *International Journal of Human-Computer Studies*, 43(5-6) :907–928, November 1995. ISSN 10715819. doi : 10.1006/ijhc.1995.1081. URL http://linkinghub.elsevier.com/retrieve/doi/10.1006/ijhc.1995.1081.

[45] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. An ontology-based context model in intelligent environments. In *Proceedings of communication networks and distributed systems modeling and simulation conference*, volume 2004, pages 270–275, 2004.

[46] Mario Gutierrez, Frederic Vexo, and Daniel Thalmann. Semantics-based representation of virtual environments. *International Journal of Computer Applications in Technology*, 23(2/3/4) :229, 2005. ISSN 0952-8091. doi : 10.1504/IJCAT.2005.006484. URL http://www.inderscience.com/link.php?id=6484.

[47] H.Dieterich, U.Malinowski, T.Küme, and M.Schneider-Hufschmidt. *State of the art in adaptive user interfaces*, pages 13–48. 1993.

[48] S Henninger. The knowledge acquisition trap. In *Proceedings of the IEEE Workshop on Applying Artificial Intelligence to Software Problems : Assessing Promises and Pitfalls (CAIA-92)*, pages 51–57, 1992.

[49] Sylvia Irawati, D. Calderón, and Heedong Ko. Semantic 3D object manipulation using object ontology in multimodal interaction framework. In *Proceedings of the 2005 international conference on Augmented tele-existence*, pages 35–39. ACM, 2005. ISBN 0473106574. URL http://portal.acm.org/citation.cfm?id=1152407.

[50] Seiie Jang and Woontack Woo. Ubi-ucam : a unified context-aware application model. In *Modeling and Using Context*, pages 178–189. Springer, 2003.

[51] A Kabbaj, K Bouzouba, and A Soudi. Amine Platform [1] : an Artificial Intelligence Environment For the Development of Intelligent Systems. *Artificial Intelligence*, 2005.

[52] Adil Kabbaj. Development of intelligent systems and multi-agents systems with amine platform. In *Conceptual Structures : Inspiration and Application*, pages 286–299. Springer, 2006.

[53] J.H. Kim. An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10) :961–973,

1. amine-platform.sourceforge.net

1999. ISSN 01628828. doi : 10.1109/34.799904. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=799904.

[54] Fabrice Lauri. Introduction à l'intelligence artificielle. http://www.site-naheulbeuk.com/utbm/IA412CM05.pdf, 2005.

[55] S Lee, Youngho Lee, Seiie Jang, and Woontack Woo. vr-UCAM : Unified context-aware application module for virtual reality. *Conference on Artificial Reality*, 2004. URL http://old.uvr.gist.ac.kr/project/2004/kist04/vr-UCAM(icat04-abstract-final).pdf.

[56] Jean-luc Lugrin and Marc Cavazza. Making Sense of Virtual Environments : Action Representation , Grounding and Common Sense. *Architecture*, 2007.

[57] John McCarthy and Buvac Sasa. Formalizing Context (Expanded Notes), 1997.

[58] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12) :1321–1329, 1994.

[59] Mohammed Nasri, Adil Kabbaj, and Karim Bouzoubaa. Integration of the controlled language ace to the amine platform. In Simon Andrews, Simon Polovina, Richard Hill, and Babak Akhgar, editors, *Conceptual Structures for Discovering Knowledge*, volume 6828 of *Lecture Notes in Computer Science*, pages 159–172. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-22687-8. doi : 10.1007/978-3-642-22688-5_12. URL http://dx.doi.org/10.1007/978-3-642-22688-5_12.

[60] Donald A. Norman. Affordance, conventions, and design. *interactions*, 6 (3) :38–43, May 1999. ISSN 1072-5520. doi : 10.1145/301153.301168. URL http://doi.acm.org/10.1145/301153.301168.

[61] J Octavia, K Coninx, and C Raymaekers. Enhancing User Interaction in Virtual Environments through Adaptive Personalized 3D Interaction Techniques. In *UMAP*, 2010. URL http://www.springerlink.com/index/N666344741624741.pdf.

[62] Johanna Renny Octavia, Lode Vanacken, Chris Raymaekers, Karin Coninx, and Eddy Flerackers. Facilitating adaptation in virtual environments using a context-aware model-based design process. In *Task Models and Diagrams for User Interface Design*, pages 58–71. Springer, 2010.

[63] Johanna Renny Octavia, Chris Raymaekers, and Karin Coninx. Adaptation in virtual environments : conceptual framework and user models. *Multimedia Tools and Applications*, 54(1) :121–142, 2011.

[64] Sejin Oh, Youngho Lee, and Woontack Woo. vr-ucam2. 0 : A unified context-aware application model for virtual environments. *ubiCNS, ProceedingCD*, 2005.

[65] Jeff Orkin. Three States and a Plan : The A.I . of F. E.A.R . In *Game Developpers Conference*, pages 1–18, 2006.

[66] Martjin Otterlo. *The logic of adaptive behavior*. 2009.

[67] Nassima Ouramdane, Samir Otmane, Malik Mallem, et al. Interaction 3d en réalité virtuelle-etat de l'art. *Revue Technique et Science Informatiques*, 28(8) : 1017–1049, 2009.

[68] Alexandros Paramythis. *Adaptive Systems : Development , Evaluation and Evolution*. PhD thesis, 2009.

[69] Mikko Perttunen, Jukka Riekki, and Ora Lassila. Context Representation and Reasoning in Pervasive Computing : a Review. *International Journal*, 4(4) :1–28, 2009.

[70] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and T. Ichikawa. The go-go interaction technique : non-linear mapping for direct manipulation in VR. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80. ACM, 1996. ISBN 0897917987. URL http://portal.acm.org/citation.cfm?id=237102.

[71] Davy Preuveneers. Support for context-driven applications in Ambient Intelligence environments. 2009.

[72] Anand Ranganathan and Roy H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6) :353–364, December 2003. ISSN 1617-4909. doi : 10.1007/s00779-003-0251-x. URL http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s00779-003-0251-x.

[73] Daniele Riboni and Claudio Bettini. OWL 2 modeling and reasoning with complex human activities. *Pervasive and Mobile Computing*, pages 1–17, February 2011. ISSN 15741192. doi : 10.1016/j.pmcj.2011.02.001. URL http://linkinghub.elsevier.com/retrieve/pii/S1574119211000265.

[74] Stuart Jonathan Russell and Peter Norvig. Introduction à l'intelligence artificielle, cours et formation en ligne des auteurs de [75]. https://www.ai-class.com/, 2011.

[75] Stuart Jonathan Russell, Peter Norvig, Ernest Davis, Stuart Jonathan Russell, and Stuart Jonathan Russell. *Artificial intelligence : a modern approach*, volume 3. Prentice hall Englewood Cliffs, 2010.

[76] Harvey Sacks. On doing "being ordinary". *Structures of social action : Studies in conversation analysis*, pages 413–429, 1984.

[77] Henrik Schärfe and Peter Øhrstrøm. Representing time and modality in narratives with conceptual graphs. In *Conceptual structures for knowledge creation and communication*, pages 201–214. Springer, 2003.

[78] Ashima Shah, Richard Hill, and Simon Polovina. A Practical Exploration of Ontology interoperability with Conceptual Graphs for added expressivity in the Semantic Web. *SPARK - The ACES Journal of Postgraduate Research*, (Cl), 2010.

[79] By Soo, Jeong Kim, Kyung Jun Gil, Hyungseok Kim, Sang Boem Lim, and Jee-in Kim. Adaptive interactions in shared virtual envrionments for heteogeneous devices. *Computer animation & virtual worlds*, pages 531–543, 2010. doi : 10.1002/cav.

[80] John F Sowa. Conceptual structures : information processing in mind and machine. 1983.

[81] John F Sowa. *Conceptual Graphs*, pages 213–217. Elsevier, 2008.

[82] Rasmus Stenholt. Efficient selection of multiple objects on a large scale. In *Proceedings of the 18th ACM symposium on Virtual reality software and technology*, VRST '12, pages 105–112, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1469-5. doi : 10.1145/2407336.2407357. URL http://doi.acm.org/10.1145/2407336.2407357.

[83] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *Workshop Proceedings*, 2004.

[84] Franck Tarpin-bernard. Interaction homme-machine adaptative. Habilitation à diriger des recherches de l'Université Claude Bernard Lyon I, 2006.

[85] David Thevenin. Adaptation en Interaction Homme-Machine : le cas de la Plasticité. 2001.

[86] Indira Mouttapa Thouvenin. Interaction et connaissance : construction d'une expérience dans le monde virtuel. Habilitation à diriger des recherches de l'université de Compiègne, 2009.

[87] Jacques Tisseau. Réalité virtuelle : autonomie in virtuo. Habilitation à diriger des recherches de l'université de Renne, 2001.

[88] Jacques Tisseau. Virtual reality and complexity. *Manifeste scientifique de l'Ecole Nationale d'Ingénieurs de Brest*, 2004.

[89] Jacques Tisseau and Marc Parenthoën. Modélisation énactive et autonomisation. In *Cahiers du colloque Intelligence de la complexité : épistémologie et pragmatique, chapitre*, volume 5, 2005.

[90] P Totterdell and P Rautenbach. Adaptation as a problem of design. *Adaptive user interfaces*, pages 61–84, 1990.

[91] Wikipedia. Description de la wiimote. http://fr.wikipedia.org/wiki/Wiimote, 2006.

[92] Wikipedia. Description de la kinect. http://fr.wikipedia.org/wiki/Kinect, 2010.

[93] Wikipedia. Description des google glass. http://en.wikipedia.org/wiki/Google_Glass, 2013.

[94] Chadwick A Wingrave, Doug A Bowman, and Naren Ramakrishnan. Towards Preferences in Virtual Environment Interfaces. *Interfaces*, 2002.

[95] Jun Yang, Hong Lu, Zhigang Liu, and Péter Pál Boda. *Physical Activity Recognition withMobile Phones : Challenges, Methods, and Applications*, pages 185–212. 2010.