Suivi d'activités et assistance cognitive ubiquitaire fondé sur les interfaces tangibles : réalisation d'un framework et d'un prototype

par

Baptiste Boussemart

mémoire présenté au Département d'Informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, juin 2007

Your file Votre référence
ISBN: 978-0-494-49465-3
Our file Notre référence
ISBN: 978-0-494-49465-3

Le 12 décembre 2007

le jury a accepté le mémoire de M. Baptiste Boussemart dans sa version finale.

Membres du jury

M. Sylvain Giroux
Directeur
Département d'informatique

Mme Hélène Pigot
Membre
Département d'informatique

M. André Mayers
Président-rapporteur
Département d'informatique

Je dédicace ce mémoire à mes parents.

Sommaire

L'assistance cognitive consiste à aider les personnes ayant des déficits cognitifs, comme les personnes atteintes de la maladie d'Alzheimer ou les traumatisés crâniens, à réaliser leurs activités de la vie quotidienne grâce à des solutions adaptées. Ces solutions peuvent être diverses. Notre approche repose sur les interfaces tangibles afin de mieux intégrer les systèmes d'assistance cognitive dans l'habitat devenu intelligent des personnes atteintes de déficits cognitifs.

En effet, les ordinateurs communiquent de nos jours avec les hommes par le biais d'interfaces graphiques. Or, ces interfaces sont peu adaptées aux personnes ayant des déficits cognitifs. De nouvelles interfaces émergent actuellement et permettent de les remplacer par des objets de la vie quotidienne. Les interfaces tangibles et les interfaces ambiantes peuvent aider à rendre l'informatique plus accessible et plus contextuelle aux personnes ayant des déficits cognitifs et à rendre par la même occasion l'assistance cognitive plus efficace.

Les interfaces tangibles permettent de manipuler des données virtuelles à l'aide d'objets réels , contrairement aux interfaces graphiques. Alors que les interfaces graphiques utilisent le clavier, la souris et l'écran pour toutes les manipulations, les interfaces tangibles, par exemple, proposent de manipuler un objet réel pour modifier une variable virtuelle. Par exemple, l'utilisateur peut détruire une variable virtuelle en plaçant l'objet réel à la corbeille. La qualité d'une interface tangible dépend de la ressemblance entre la manipulation réelle et la manipulation virtuelle. Il devient alors possible de supprimer le clavier et la souris. Les interfaces ambiantes, qui ne sont pas traitées dans ce mémoire, permettent de leur côté de supprimer l'écran.

Ce mémoire propose une architecture centrée sur le modèle *Token And Constraint* (TAC). Le modèle TAC est le modèle qui répond le mieux à la définition d'interactions tangibles pour l'assistance cognitive. Il a été appliqué à la réalisation d'une assistance simple dans la cuisine. Aussi pour construire l'architecture et le cadre de programmation,

deux applications ont été réalisées.

La première application utilise des interactions tangibles de préparation d'une recette de cuisine de manière virtuelle. Des objets posés sur une table sont identifiables par une caméra vidéo, grâce à une forme unique apposée sur chacun de ces objets. Des informations digitales et des zones sont projetées sur la table. Lorsqu'un objet entre dans une zone, une certaine action est réalisée sur la variable associée à l'objet, par exemple la casserole. Les actions définies par chacune de ces zones sont directement liées aux actions de la vie quotidienne pour la réalisation d'une recette de cuisine, par exemple : couper, mélanger, verser, etc. L'ensemble des variables associées aux objets définissent un modèle de la recette.

La seconde application utilise des interactions tangibles dans une vraie cuisine. La technologie RFID (Radio Frequency Identification Device) permet dans ce cas d'identifier les objets réels. Des étiquettes RFID sont collés sur les objets, tels que la boîte de riz, le verre d'eau, etc. Les lecteurs de tags RFID détectent la présence d'étiquettes RFID dans certaines zones ; le placard, l'évier et la cuisinière. Les interactions TAC sont alors utilisées pour interpréter la présence des objets correspondant aux étiquettes dans ces zones en fonction de la recette à réaliser.

Remerciements

Je remercie de tout mon cœur mon directeur de Maîtrise M. Sylvain Giroux. Travailler avec lui a été très enrichissant, une expérience qui restera importante pour moi. Le travail n'a pas toujours été facile et c'est grâce à son soutien et à ses commentaires que ces travaux ont pu aboutir.

Surtout je remercie mes parents pour tout leur amour et leurs soutiens. Je n'ai pas toujours été facile et je veux tourner ce chapitre en leur disant que je les aime beaucoup aussi.

Je remercie les membres du Jury, Mme. Hélène Pigot et M. André Mayers, qui ont bien voulu passer du temps à corriger et commenter mon mémoire. Des choses me sont apparues différentes et je suis sûr que le contenu du mémoire en sera amélioré. Je suis admiratif par la quantité de choses que nos directeurs et professeurs supportent durant une année : je remercie donc tous les professeurs de l'Université de Sherbrooke, qui sont très disponibles.

Enfin, je veux remercier toutes les personnes que j'ai côtoyées pendant cette Maîtrise. Je suis content d'avoir travaillé dans le laboratoire DOMUS dans un cadre très convivial. Je leur souhaite à tous bonne chance.

Je remercie le Canada et l'Université de Sherbrooke de m'avoir accueilli. Un remerciement particulier à Mme. Lynn Le Brun, pour les dernières démarches qui ont été hâtives.

Je remercie la France pour son soutien financier pendant mes six années supérieures. Ils seront contents de me voir revenir.

Enfin un grand merci à l'ESÉO pour l'opportunité de travailler à l'étranger. C'est une chance pour tous ceux qui ont participé à cette aventure, car nous faisons partie des premiers à bénéficier de cela.

Table des matières

Liste des tableaux

Liste des figures

Introduction

De plus en plus de personnes ont des difficultés cognitives au sein de nos sociétés actuelles, qu'elles soient schizophrènes, traumatisées crâniens, ou atteintes de la maladie d'Alzheimer. De plus en Occident, les courbes démographiques prévoient un vieillissement d'une grande proportion de la population [12]. De ce fait, un plus grand nombre de personnes seront atteintes de la maladie d'Alzheimer. Ces personnes auront des difficultés à exécuter, à divers degrés, les activités de la vie quotidienne. Cependant pour compléter, voire suppléer le manque d'aidants, il est possible d'utiliser les technologies de l'information, de l'automatisation et de la robotique [44]. C'est donc un marché énorme qui s'ouvre à présent. Auparavant, la domotique n'a pas su répondre aux demandes de ce marché, aujourd'hui beaucoup de réponses voient le jour.

Par la voie de la recherche dans les domaines des technologies et des sciences cognitives, un système intelligent pourrait être déployé dans une résidence afin d'aider son occupant et exiger ainsi très peu, voire aucune aide extérieure.

Les systèmes informatiques actuels utilisent des interfaces graphiques, reposant sur la combinaison clavier, souris et écran. Ces interfaces permettent la construction d'une gamme importante d'applications, car elles offrent des variations infinies : grands écrans pour afficher un grand nombre d'informations, clavier pour taper du texte et des commandes, et la souris pour pointer des objets virtuels affichés sur l'écran. Tous les postes informatiques utilisent ces interfaces. Par contre, de telles interfaces ne sont pas adaptées aux personnes ayant des déficits cognitifs, ni même pour des utilisateurs occasionnels car l'apprentissage est fastidieux [64].

Comment concevoir un assistant basé sur des interfaces informatiques "traditionnelles" si elles sont mal adaptées pour ces personnes ? Vous l'aurez compris, c'est impossible.

La recherche est allée plus loin depuis quelque temps en ramenant ces interfaces vers des objets tangibles. Mais elles ne sont pas aussi facilement personnalisables que les

interfaces graphiques communément utilisées.

Mon travail a consisté à identifier et construire des interfaces tangibles adaptées pour l'assistance cognitive personnalisée. De plus, les assistants actuels consiste à entrer à la main les actions réalisées sur un logiciel avec PDA (Personal Digital Assistant) (avec un écran et clavier) et qu'ainsi le renseignement de l'assistant (le système d'information) n'est pas contrôlé par les objets physiques du monde réel. Mon travail a permis d'avancer dans la résolution de ce problème. Mais tout d'abord, voyons ce que réalise le laboratoire DOMUS afin de mieux comprendre comment s'intègre ce projet dans l'ensemble des travaux en cours.

Le laboratoire DOMUS

Le laboratoire DOMUS a pour vocation de proposer un habitat intelligent permettant aux personnes ayant des déficits cognitifs de garder leur indépendance. DOMUS veut répondre à la problématique de façon simple, efficace, transparente, économique et éthique. Le laboratoire ne cherche pas à proposer une solution où toutes les tâches, que la personne doit accomplir, sont automatisées et où la personne deviendra de fait dépendante de la technologie. Ainsi, il repousse à ce jour l'automatisation.

Dans le même esprit, la surveillance par caméra soulève des problèmes éthiques et est écartée des solutions envisagées. La problématique ci-dessous se veut être un cahier des charges d'une aide technologique humaine.

La problématique

Beaucoup de laboratoires conçoivent des systèmes informatiques pour assister la personne ayant des déficiences cognitives, afin de promouvoir l'autonomie des personnes dont les capacités cognitives sont affaiblies. Chaque laboratoire possède ses propres compétences, ses propres visions pour l'avenir et donc leurs méthodologies sont différentes. Quelle est la méthodologie du laboratoire DOMUS pour concevoir cet assistant ?

DOMUS veut concevoir un système dans la lignée du concept de l'intelligence ambiante (AmI[1]) [2]. L'ambiance intelligente recherchée implique de multiples propriétés et défis dans la conception de ce système.

[1] Ambient Intelligence

- Premièrement le système doit être diffus dans l'environnement [75], ce qui implique la conception de multiples sous-systèmes mis en réseau.
- Deuxièmement le système est intelligent et contextuel, cela entend que le système répond à l'usager de manière intelligente, c'est-à-dire qu'un humain ne fait pas la différence entre la réponse d'un humain ou d'une machine (test de Turing) [38],
- Troisièmement, l'intégration dans l'habitat et l'utilisation par l'usager doit être naturelle et transparente.

Le système devient plus fonctionnel et plus efficace, mais il est en contrepartie beaucoup plus complexe à concevoir, à installer et à maintenir. Le système vit comme la maison et la vie de ses usagers, donc il doit être capable de s'adapter, de s'étendre, et de se déplacer (migrer).

Les clientèles actuellement ciblées sont :

1. les personnes âgées atteintes de démence dont la maladie d'Alzheimer, une maladie qui détériore les tissus du cerveau,
2. les traumatisés crâniens-cérébraux (TCC), un traumatisme causé par un choc à la tête, souvent dû à un accident de voiture.
3. la schizophrénie,
4. les personnes atteintes de déficiences intellectuelles.

Une solution simple et commune aux quatre pathologies est-elle envisageable ? Bien que chaque catégorie ait ses propres spécificités, plusieurs points sont similaires au niveau des types de déficits. Les déficits assistés par DOMUS sont la planification, la mémoire, l'attention et l'initiation.

De plus, les interfaces peuvent varier considérablement d'une clientèle à l'autre. Les concepteurs d'interfaces doivent être vigilants à ce que le produit réponde aux besoins d'une population qui n'est peut-être pas habituée à l'informatique. Par exemple, la maladie d'Alzheimer touche principalement les personnes âgées. Les TCC sont par contre en majorité des jeunes habitués avec l'informatique et les jeux vidéos.

Finalement, ces personnes sont entourées de leurs familles et aidants. L'habitat doit être aussi assimilé, utilisable et paramétrable par les aidants, afin que le système réponde précisément aux besoins [37]. Malgré un système pour le moins complexe, la configuration du système doit être simple.

De la portée des travaux de recherche de DOMUS

Bien que l'objectif premier du laboratoire DOMUS soit la conception de systèmes d'information qui assistent les personnes avec des déficiences cognitives qui soient bien adaptée à ces personnes, la portée de ses travaux de DOMUS dépasse la seule assistance cognitive. Tous les concepts développés ici pourront servir hors du sujet de cette Maîtrise. Le laboratoire travaille sur beaucoup de champs comme les programmes adaptables, l'auto configuration des systèmes, les systèmes répartis. L'utilisation de cette assistance pourrait être aussi utile les personnes souhaitant une domotique intelligente, complétant la simple automatisation de l'habitat. Toutes les techniques en recherche au sein de DOMUS pourront apporter des idées à l'informatique en général.

Les expérimentations des solutions proposées par DOMUS

DOMUS possède son propre appartement permettant d'expérimenter les solutions conçues. Des travaux dans le domaine de la localisation et de la reconnaissance de tâches ont été menés pour des expérimentations dans cet appartement. Un projet d'assistance hors du domicile a été mené [47] et c'est le projet le plus finalisé de DOMUS à ce jour. Le système utilise la communication sans fil pour accéder aux services nécessaires à l'assistance, ainsi que la technologie de localisation *GPS*. Enfin de nombreuses études sont menées pour classifier les symptômes de ces personnes, soit par des observations ou par des simulations.

Les travaux de Denis Vergnes [72] et de Jérémy Bauchet [4] sont de leur côté les premiers jalons pour l'intégration d'interfaces tangibles pour l'assistance à l'intérieur.

Les interfaces tangibles au service de l'assistance cognitive

De récents travaux menés dans le domaine des interfaces homme/machine comme la réalité augmentée, les interfaces tangibles et les interfaces ambiantes, offrent de nouvelles perspectives pour les travaux menés au sein du laboratoire DOMUS. Nos travaux ont pour but d'explorer dans quelle mesure ces interfaces émergentes peuvent rendre l'assistance plus performante et plus compréhensible pour les personnes ayant des déficits cognitifs.

Nos recommandations seront basées sur les recherches menées en cognition et en

interfaces homme/machine.

D'un côté, la conception des interfaces a fait couler beaucoup d'encre depuis plusieurs décennies. Il est question de technique, de convivialité, de performance, de nouveautés et de conception.

La réalité augmentée, les interfaces tangibles et les interfaces ambiantes ont émergé depuis la fin des années 90 comme de nouvelles approches prometteuses afin de rendre les interfaces plus fonctionnelles et plus intelligentes pour certains domaines précis.

De l'autre côté, l'assistance cognitive est liée aux sciences cognitives afin de pallier aux déficits cognitifs de planification, de mémorisation, de prise de décisions et de réflexion. Les sciences cognitives étudient le fonctionnement du cerveau, proposant plusieurs modèles pour comprendre le comportement humain. Au niveau de l'assistance, la prise en compte des besoins de la personne est cruciale. Ainsi, l'assistance proposée doit être facilement comprise et utilisée. La personnalisation de l'assistant est une fonction cruciale (cf. 1.1.2).

Ces points sont approfondis dans le chapitre 1. Le chapitre 2 présente le cas d'étude qui donnera lieu au prototype.

Intégration avec les problématiques de DOMUS

Le laboratoire DOMUS a une logique d'intégration des projets afin de répondre à son objectif. Voici en quelques phrases comment s'intègre mon projet avec les autres.

Informatique diffuse

Mon prototype devra s'intégrer dans l'environnement diffus construit au laboratoire et le compléter. Le laboratoire a intégré des microphones, des capteurs de débit d'eau, des capteurs de présence, tels des tapis tactiles, des capteurs infrarouges et des capteurs de détection d'ouverture de portes.

Mon projet étend cette infrastructure en utilisant la technologie *RFID* [2] pour identifier et localiser, une fonctionnalité nécessaire pour réaliser des interfaces tangibles.

Localisation

Les capteurs de présence servent à la localisation des personnes. Les informations fournies par ces capteurs sont traitées afin de leur donner une sémantique intéressante

[2] Radio Frequency Identification Device

pour l'assistance cognitive.

La technologie *RFID* permet aussi de localiser les personnes, mais elle est surtout utilisée pour localiser les objets. Des étiquettes RFID sont placées sur les objets. Ces étiquettes sont lues par des bornes d'identification *RFID*. Il devient ainsi possible de localiser les objets, à partir des positions des lecteurs RFID et de leur portée.

Contexte et description de l'environnement

Il faut savoir qu'une information n'a pas de sens prise isolément, il faut toujours la placer dans un contexte. Ainsi, lorsqu'un capteur envoie une information, le système interprète cette information en fonction du contexte. Pour donner du sens à cette information, l'environnement doit donc être décrit. Il faut aussi donner une description de ce capteur et de l'information qu'il fournit.

Beaucoup d'interprétations de l'information sont possibles et les décrire dans l'environnement est très fastidieux. La définition de contextes permet de subdiviser, de classifier les situations. Par exemple, on interprétera différemment la sortie d'une personne si c'est l'heure de son rendez-vous chez le dentiste ou s'il est tard la nuit. Dans le premier cas, on lui rappellera de prendre son téléphone portable. Dans le second cas, une alarme sera déclenchée.

La reconnaissance d'activités est nécessaire. À cette fin, mon projet décrit ce que les bornes *RFID* doivent détecter et le sens qu'il faudra donner aux informations récoltées. Le modèle *Token And Constraint* sera utilisé comme référence pour ce faire et pour la construction des interfaces tangibles pour l'assistance cognitive.

Reconnaissance d'activités

La reconnaissance d'activités a été traitée par plusieurs étudiants du laboratoire DOMUS. Il s'agit de reconnaître des patrons liés à des activités dans le flot d'informations provenant des capteurs. C'est un des piliers de la conception de l'assistance que DOMUS veut construire. La technologie RFID, en permettant la localisation des objets, donne des informations plus riches à la reconnaissance d'activités.

Architectures logicielles et réseaux

L'approche visée pour DOMUS est tout sauf centralisée, mais de multiples ordinateurs, qu'il s'agisse de processeurs embarqués ou de serveurs, sont interconnectés entre

eux pour constituer un assistant cognitif intelligent. Un tel système est difficile à concevoir, à configurer et à maintenir, sans parler des nombreuses fautes et erreurs qui peuvent survenir.

Pour ce faire, DOMUS se dirige vers une architecture *OSGi* (*Open Services Gateway initiative*), qui est modulaire et configurable. L'architecture de mon prototype devra être transformée en service, ou agent, de pair-à-pair.

Appareils mobiles

L'utilisation des appareils mobiles, comme les téléphones cellulaires, les montres ou les *PDA* (*Personal Digital Assistant*), ouvrent aussi de nouvelles perspectives pour l'assistance cognitive.

Mon projet n'utilise pas d'appareils mobiles. Cependant, un PDA pourrait servir à lire les tags *RFID* d'une part et apporter une assistance d'autre part.

Objectifs et démarche

Nos visées sont de natures prospectives à l'intérieur de laboratoire DOMUS. Il s'agit d'explorer l'utilisation des interfaces émergentes (les interfaces tangibles, les interfaces ambiantes, etc.) pour la réalisation de systèmes d'assistance cognitive. Pour ce faire, nous avons d'abord examiné diverses applications et modèles reposant sur les interfaces émergentes. Nous avons aussi examiné les contraintes posées par les déficits cognitifs sur les interfaces homme/machine. Ensuite nous avons conçu et implémenté deux prototypes d'assistant cognitif pour la réalisation d'une recette de cuisine afin d'une part de voir les possibilités affectés par les interfaces émergentes et d'autre part de préparer l'intégration de ces interfaces dans l'infrastructure de DOMUS et de futurs projets de recherche. Finalement, ces prototypes permettent de mesurer l'efficacité, la pertinence, la convivialité et la performance en termes d'assistance cognitive et de contraintes informatiques.

Structure du mémoire

Ce mémoire présente donc un assistant cognitif reposant sur les interfaces tangibles Le chapitre 1 développe les connaissances dans les domaines des interfaces tangibles et de l'assistance cognitive. Le chapitre 2 présente l'application des interfaces tangibles pour l'assistance cognitive. Ensuite les aspects techniques utilisés sont détaillés dans le chapitre 3. En possession de ces connaissances, les solutions envisagées sont expliquées

dans le chapitre 4. Enfin, la réalisation du projet ainsi que les résultats sont livrés dans le chapitre 5.

Chapitre 1

Les connaissances pour l'assistance cognitive et les interfaces tangibles

Ce chapitre présente l'état des connaissances dans les domaines principaux de ce mémoire : l'assistance cognitive (section 1.1) et les interfaces tangibles (section 1.3).

La première partie traite de l'assistance cognitive (section 1.1) afin de déterminer comment concevoir une assistance appropriée. Des solutions déjà existantes, des idées et des recommandations de la communauté scientifique sont répertoriées.

La deuxième partie étudie les interfaces homme/machine (section 1.2) et leur conception. C'est un champ de recherche à part entière. En particulier, nous nous demandons quels sont les besoins spécifiques en cette matière pour l'assistance cognitive ?

La troisième partie aborde les interfaces tangibles (section 1.3). Divers modèles et réalisations d'interfaces sont comparés avec les interfaces tangibles afin d'en montrer les forces ou les faiblesses. Ainsi, d'autres modalités d'interfaces peuvent apporter des fonctionnalités intéressantes et complémentaires aux interfaces tangibles.

Pour terminer, des projets participant des interfaces tangibles et de l'assistance cognitive sont présentés (section 1.4).

1.1 Assistance Cognitive

La cognition est la fonction mentale permettant d'acquérir et de mettre en œuvre des connaissances. Cela met en jeu les facultés suivantes : la planification, la mémorisation, l'évaluation, la résolution de problème et la prise de décisions.

L'assistance cognitive permet de suppléer les capacités cognitives différentes d'une personne. L'assistance cognitive peut prendre plusieurs formes. Par exemple, un système de rappel des activités à réaliser, un nœud dans un mouchoir, etc. L'assistance cognitive peut prendre que nous dévelepperons reposera sur la reconnaissance d'activités et sur l'assistance procédurale

1.1.1 Problèmes cognitifs

Dans cette section, nous montrerons quatre grandes catégories de problèmes cognitifs et les façons de les assister : les problèmes d'attention, les problèmes de planification, les problèmes d'initiation et les problèmes de langage.

Les problèmes d'attention

Les systèmes informatiques, comme l'assistant cognitif dont il est question, reposent sur l'attention pour interagir avec les utilisateurs. Cela peut poser des problèmes importants pour les personnes ayant des déficits cognitifs, mais cette prise d'attention est nécessaire dans bien des cas.

L'assistance captera l'attention à l'aide de dispositifs lumineux ou sonores disposés aux endroits où l'attention de l'utilisateur peut facilement être sollicitée [45]. Dans un cas d'urgence, tel un danger d'incendie, les distractions ne peuvent pas être tolérées. Le système captera l'attention de la personne, et si rien ne se résout rapidement, les urgences seront appelées automatiquement. Cependant, il faut solliciter l'attention des personnes de manière judicieuse et adaptée. Car de telles sollicitations peuvent entraîner des erreurs et des changements de comportement chez la personne. En particulier, on risque de forcer la personne à traiter plusieurs tâches simultanément, et le multitâche est difficile à accomplir tant il sollicite la planification, la gestion de temps, etc. [45], des ressources fortement limitées chez les personnes ayant des déficits cognitifs. L'attention est donc un aspect crucial qu'il faut gérer de manière appropriée.

Quelles solutions faut-il alors envisager ? L'informatique diffuse a beaucoup travaillé

sur cette question. C'est là que les domaines de l'assistance cognitive, de l'informatique diffuse et des interfaces homme/machine se rejoignent [45] [41]. Trois niveaux ont été proposés pour analyser les informations à afficher : l'abstraction de l'information, la notification et la transition.

L'abstraction s'applique lorsque le système ne devrait pas donner l'information telle quelle, mais plutôt lorsque le système devrait fournir une information équivalente, qui sera interprétée de la même manière. Cela permet, en lien avec la notification et la transition, d'utiliser d'autres types d'interfaces qu'un écran pour afficher du texte. On peut alors utiliser des couleurs, des images, diffuser des mélodies, etc. Le niveau de notification est aussi un facteur important, car certaines informations ont moins d'importance que d'autres. Dans ce contexte, l'utilisation de métaphores peut être significative et puissante, si ces métaphores sont bien comprises par l'usager.

Il faut savoir distinguer une information urgente d'une information qui n'a pas d'importance comme une mise à jour de la météo. Pour cela, les niveaux d'interruption définissent comment la transition se fera : ignorer, changer d'ambiance, avertir, interrompre et demander une action. Enfin, la transition définit comment l'information est diffusée : de manière brutale ou de manière douce, en clignotant ou en continu. À partir du travail décrit dans [41], une boîte à outils a été proposée. Il est donc intéressant de s'en inspirer. Les éléments les plus significatifs pour mettre en oeuvre ce travail sont la définition des affichages et d'un plan de transit de l'information avec l'application de filtres. Ce système propose un affichage ambiant qui respecte l'attention de l'utilisateur en terme d'abstraction, de notification et de transition.

Les problèmes de planification

La mémoire prospective stocke les actions planifiées. Un indice pourra alors rappeler que telle ou telle action est à réaliser. Des études montrent qu'il est possible d'apporter des solutions à ces problèmes [60]. L'utilisation d'un agenda constitue souvent une solution simple pour les personnes ayant des difficultés à planifier leurs tâches [17]. Malheureusement, les solutions manquent de dynamisme et ne sont pas adaptées pour les tâches de la vie quotidienne. C'est pourquoi la réalisation de plusieurs activités concurrentes est très peu supportée dans les assistants actuels. Les compétences nécessaires de la part des usagers sont la planification, la prise en compte du temps, l'auto-critique et l'auto-contrôle de leurs tâches.

Les problèmes d'initiation

Quels sont les stimuli qui vont permettre à une personne de se rappeler une tâche ? A-t-elle envie de faire la tâche ? L'initiation est l'acte de commencer une tâche de la vie quotidienne. Chez les personnes atteintes de déficits cognitifs, il arrive fréquemment qu'elles n'arrivent pas à se décider à entamer la réalisation d'une tâche, soit qu'elles ne savent pas comment commencer, soit qu'elles ont peur de ne pas pouvoir juger de l'ampleur, de la faisabilité ou du succès de la tâche. L'assistant cognitif doit aider les personnes à être motivées et confiantes.

En outre, il existe un lien entre planification et initiation. Par exemple, le modèle de Kang et Yong promeut l'exploration de l'interface homme/machine par l'utilisateur [30]. Ce modèle requiert des compétences en initiation et planification. En effet, l'exploration nécessite une curiosité et donc une initiation à l'exploration. L'exploration doit être planifiée. Dans une interface web par exemple, la planification est fortement sollicitée pour que la recherche soit bien gouvernée. Kang propose un langage graphique pour représenter les différentes opportunités que peut suivre la personne dans sa navigation dans l'interface. Cet outil permet l'analyse de l'interface pour qu'elle soit facilement explorée et apprise.

Les problèmes de langage

Des délais de compréhension sont remarqués avec le vieillissement [17]. De plus, la quantité d'information traitée en parallèle devient plus faible avec des personnes de plus de soixante-cinq ans. Les phrases deviennent plus simples. La compréhension de phrases complexes cause une anxiété chez ces personnes [64]. Il est toutefois intéressant de noter que le temps de lecture est plus important sur un écran, mais que la compréhension du texte reste la même [64].

Le langage devient donc plus simple. C'est pourquoi Hendler [18] énonce sept règles pour qu'un langage d'interaction soit une réussite :

1. le langage doit prendre peu de temps de calcul sur une machine,
2. le langage doit être facile et rapidement appris,
3. la portée et les possibilités du langage doivent être apparentes lors de sa lecture,
4. une fois appris, le langage doit être très versatile,
5. le langage ne doit pas être prévu en fonction de l'expérience des usagers,

6. le langage ne doit pas être oublié dans le long terme,
7. le langage ne doit pas prendre de trop de temps quant à la réalisation des tâches avec celui-ci.

Ces éléments sont importants si un langage d'interaction avec la machine, notamment iconique, est utilisé [42]. Le langage iconique peut s'avérer un moyen de communication viable puisque l'image est mieux traitée par le cerveau que le texte et qu'il faut que la personne puisse donner des ordres à la machine. Des travaux ont été menés en ce sens comme *VilAug* [71]. Un langage iconique simple comme langage d'interaction avec la machine est assurément une avenue prometteuse. Le projet *GALOP* travaille dans cette veine [66].

Les conséquences sur la personnalité

Du stress et de l'anxiété peuvent être causés par les déficits cognitifs et par une mauvaise assistance [64] [17] [45]. Des comportements antisociaux peuvent alors en découler. Assister une personne devient alors complexe, c'est pourquoi il faudrait prendre en compte son profil et son historique afin de ne pas la perturber. L'enjeu est de concevoir des interfaces homme/machine suffisamment conviviales pour que la personne ait envie d'utiliser l'assistance informatisée proposée.

La motivation est un facteur très important dans le succès de l'assistance cognitive et déterminante dans la qualité de vie de la personne. La motivation est reliée à la satisfaction des besoins et au fait que la personne les perçoive comme plus facilement réalisables. La pyramide de Maslow distingue cinq catégories de désirs : besoins physiques, besoins de sécurité, besoins sociaux, besoins d'estime et l'accomplissement personnel (c'est-à-dire rendre concret ses désirs)[1] [39].

Les personnes qui ne sont pas motivées ont tendance à exprimer des comportements antisociaux. La motivation est donc un aspect très important à surveiller. Un assistant cognitif peut permettre aux gens d'accomplir des choses jusqu'alors difficiles à atteindre et la motivation de ces personnes augmentera avec l'atteinte des résultats escomptés.

1.1.2 Vers des assistants cognitifs

Les assistants sensibles au contexte utilisent les technologies d'intelligence artificielle. Les interfaces procédurales facilitent souvent l'assistance aux personnes [60] [17] [45] [28].

[1] http ://en.wikipedia.org/wiki/Maslow's_hierarchy

Une interface procédurale [6] suit l'utilisateur étape par étape pour la réalisation d'une procédure donnée, par exemple une tâche de la vie quotidienne. Il ne peut pas se tromper normalement. L'assistant laisse peu de liberté à l'usager, mais la probabilité que la tâche soit réalisée est meilleure [6]. La combinaison de l'intelligence artificielle, de la sensibilité au contexte et de l'assistance procédurale, permet de concevoir et réaliser des assistants capables d'anticiper, d'interagir et de récupérer des situations problématiques. Nous appellerons "assistant procédural" un assistant cognitif qui utilise les interfaces procédurales [6]. Finalement afin de répondre à certaines considérations éthiques, l'assistant doit respecter les règles suivantes [6] :

1. le système ne doit pas remplacer les actions de la personne, ce qui interdit l'automatisation,
2. l'assistant doit être personnalisé par rapport au profil de l'utilisateur, afin de créer une assistance pertinente et précise,
3. le système ne doit pas être intrusif ; en particulier il faut éviter la surveillance intrusive et inutile ; ainsi cela suppose un système fermé ou avec des rapports résumés à l'essentiel à l'attention des médecins,
4. l'utilisateur peut présenter des troubles de comportement, il faut donc que le système n'induise pas encore plus de stress chez la personne,
5. le système doit proposer des tâches que la personne peut réellement réaliser.

Un assistant cognitif n'est pas systématique dans le contrôle de la tâche, contrairement à un assistant procédural, qui suit la procédure pas à pas. L'assistant cognitif détecte toutes les activités, alors que l'assistant procédural travaille avec seulement une tâche à la fois. La procédure est en général visible pour l'utilisateur dans l'assistant procédural, alors qu'elle n'apparaît pas dans l'assistant cognitif. Il est plus facile de développer un assistant procédural, qu'un assistant cognitif, car l'un est déterministe et l'autre probabiliste.

Manger, prendre ses médicaments, s'habiller, se laver, se préparer à manger, faire les courses, faire le ménage, faire le repassage, faire la lessive, avoir des loisirs, avoir des communications avec ses proches [58] [3] sont toutes des activités de la vie quotidienne qui peuvent bénéficier de l'assistance cognitive chez les personnes atteintes de déficits cognitifs.

Une étude réalisée dans la communauté, auprès des aidants naturels, des aidants professionnels et des personnes atteintes de déficits cognitifs a identifié de nombreuses pistes pour répondre à des situations actuellement problématiques chez la clientèle visée

[58]. Voici les plus importantes :

- un système d'alarme de feu qui appelle les urgences directement,
- un dispositif pour arrêter la cuisinière automatiquement,
- un dispositif de détection de robinet ouvert qui ferme automatiquement,
- un dispositif de détection du niveau de l'eau de la baignoire et des lavabos,
- une alarme pour les portes extérieures ouvertes,
- la localisation de la personne, bien qu'elle pose un problème éthique évident,
- la détection d'aliments périmés,
- la détection des chutes de la personne,
- un système rapide d'appel des aidants naturels,
- un étalon pour mesurer la quantité d'un produit qui sonne et qui parle,
- un système qui rappelle les rendez-vous,
- un système qui permet à la personne de s'orienter,
- un système de verrouillage des portes,
- un système qui rassure la personne,
- un système qui rappelle les consignes de sécurité.

1.2 Les interfaces homme/machine et l'assistance cognitive

Les interfaces homme/machine sont des moyens de communication entre l'homme et la machine. L'origine remonte au premier outil, puis à la création de la machine. En effet, l'interface entre l'outil et la main est le manche ou la poignée.

De nos jours, le bureau de travail est muni d'un microordinateur où un utilisateur interagit par le biais d'un écran, d'une souris et d'un clavier. Les études actuelles offrent des interfaces innovantes et des techniques de conception et d'évaluation. D'autres domaines, comme les jeux vidéos, ont apporté beaucoup de créativité pour les interfaces. Des chercheurs ont conçu de nouvelles interfaces pour repousser les limites : interfaces tactiles multitouches, interfaces 3D tactiles (Microsoft), etc.

Toutefois, dans un domaine aussi spécifique que l'assistance cognitive, certaines règles sont à respecter quant aux interfaces homme/machine. Concevoir des interfaces n'est pas facile. Plusieurs types d'approches s'affrontent pour savoir comment bien concevoir une interface homme/machine. La conception par participation [76] prend en compte les opi-

nions d'un ensemble d'utilisateurs, alors que la conception orientée système est focalisée sur comment bien concevoir le système sans l'avis des utilisateurs. Les projets orientés systèmes sont plus soucieux de la qualité de fonctionnement du programme et souvent il y a peu ou pas d'interactions avec l'homme. En général, les études montrent qu'il faut impliquer les utilisateurs pour que l'interface soit appréciée. Il existe une dernière méthode de conception qui est la conception orientée utilisateur, où l'interface est spécialement adaptée pour l'utilisateur [5] [46]. Cela implique, en autres, que l'interface soit personnalisable.

C'est généralement ce qui est choisi par l'intelligence ambiante, car pour que le système soit perçu comme intelligent il faut que l'interface soit comprise par l'utilisateur. Il faut aussi que l'interface fonctionne de manière fluide avec les tâches de la personne. C'est pour cela que les interfaces conçues spécifiquement pour l'utilisateur répondent à ces problèmes.

La conception orientée utilisateur est donc adaptée aux personnes ayant des déficits cognitifs [28] [49] [60]. Le profil de l'utilisateur basé sur ses forces et ses faiblesses [37] décrit ses capacités [17]. Il peut aussi préciser les préférences, les goûts et l'état de santé de l'utilisateur. L'interface est personnalisée en fonction du profil.

La personnalisation permet de réaliser des interfaces adaptables. Les interfaces adaptables changent suivant l'état du programme. En particulier, les interfaces procédurales sont des interfaces adaptables, car elles changent en fonction de l'étape de la procédure. Si le programme possède un nombre non mémorisable d'états, le système ne sera jamais adopté. C'est pourquoi les interfaces adaptables ne sont pas plébiscitées pour une utilisation par un assistant cognitif pour les personnes âgées [28]. Cependant, les assistants demeurent intéressants, si leur interface demeure compréhensible par l'utilisateur.

Les interactions doivent être simples et faciles à utiliser [60]. Plus un système a d'entrées et de sorties, plus d'états sont possibles et par conséquent plus il sera complexe [6]. Plus il sera complexe, plus il sera rejeté par les utilisateurs.

1.3 Les interfaces tangibles

Les interfaces tangibles appartiennent à une nouvelle catégorie d'interfaces.[2] Elles consistent à "augmenter" le monde physique en associant de l'information virtuelle aux

[2]Le blog *Pasta and Vinegar* reporte les nouveautés en interfaces tangibles spécifiquement : http ://tecfa.unige.ch/perso/staf/nova/blog/category/tangibleintangible/

objets physiques. L'interaction de l'utilisateur sur ces objets modifie les informations, qui sont ainsi manipulées de la même manière que des objets physiques. Il ne s'agit pas par contre de ramener les objets virtuels des interfaces graphiques à la réalité, comme les interfaces graphiques qui ont copié le bureau physique. Ainsi, construire une interface tangible à partir d'une interface graphique semble une approche incorrecte d'après la littérature.

Les interfaces tangibles permettent de supprimer la souris, le clavier et l'écran qui sont trop complexes pour les utilisateurs qui ont des handicaps [43]. Le fait d'utiliser des objets réels peut avoir une signification beaucoup plus forte que les interfaces graphiques virtuelles. Les interfaces tangibles sont donc envisagées pour supprimer les interfaces communément utilisées, c'est-à-dire la souris et le clavier. L'écran est souvent remplacé par des affichages ambiants ou des murs et des toiles interactifs.

Les interfaces tangibles sont issues des travaux dans plusieurs domaines : les interfaces homme/machine, l'informatique, le design et le multimédia interactif. De nouveaux types d'interaction apparaissent et reposent sur une conception virtuelle et physique. La "tangibilité" des interactions avec une interface tangible telle que perçue par les utilisateurs dépendra de plusieurs facteurs comme l'expertise, la culture, l'inné et les facultés sensorielles et motrices [23].

Depuis la création du concept d'interface tangible, plusieurs définitions et modèles ont été proposés. Dans ce qui suit nous verrons les modèles suivants : *graspable interface*, *MCRpd*, table interactive, *Token and Constraint* et assemblage.

La première définition était l'interface "préhensible[3]" donnée par Fitzmaurice au milieu des années 1990. Ullmer et Ishii ont beaucoup participé à la définition des interfaces tangibles les années suivantes [25], en employant les termes *tangible-bits* et *TUI*, pour *tangible user interface*. À cette époque les types d'interfaces proposées étaient le *metaDesk*, le *transBoard* et *ambientRoom* [34]. Il semble qu'*ambientRoom* ne soit plus proposé comme interface tangible, il appartient maintenant au domaine des interfaces ambiantes. De leur côté, *metaDesk* et *transBoard* sont des surfaces interactives, comme les tables interactives.

En 2001, le modèle *MCRpd* (Modèle Contrôleur Représentation physique et digitale) est proposé comme analogie au modèle *MVC* (Modèle/Vue/Contrôleur) des interfaces graphiques, avec toujours la même vision de coupler l'objet et l'information [67]. Ce modèle met en avant le fait que les interactions passent par des manipulations physiques.

[3]traduction littérale de *graspable interface*

Toutefois, une représentation des manipulations reste à faire et ce modèle semble demeurer une vue de l'esprit.

En 2003, Fishkin [14] a proposé une autre classification des interfaces tangibles et une analyse de la définition du terme interface tangible. Bien qu'il soit facile de proposer des interfaces tangibles, la "tangibilité" d'une interface reste un critère d'appréciation subjectif, déterminant toutefois pour le succès d'une interface tangible. La classification des interfaces de Fishkin se base sur l'utilisation de noms pour représenter les objets et de verbes pour représenter les actions. L'incarnation de l'interface tangible vient du fait qu'elle propose des interactions complètes, proches de la réalité, en utilisant l'environnement. L'objet peut avoir un sens littéral fort. Par exemple, un conteneur est utilisé pour transférer de l'information entre deux variables. Un jeton est un objet qui symbolise l'information. Les outils sont des objets qui servent à modifier les informations.

De son côté, Hornecker [19] [21] [20] propose en 2005 une série d'articles pour défendre sa structure d'interfaces tangibles. Selon ses travaux, la vision des interfaces tangibles doit être centrée vers les données et l'espace. L'expression du mouvement est pour la première fois définie.

Dans le reste de ce mémoire, nous utiliserons les trois catégories d'interfaces tangibles proposées par Ullmer et Ishii [68] pour situer notre travail : la table interactive, le token+constraint et l'association d'objets. Ces catégories permettent de bien classifier les projets.

Dans cette section, nous verrons comment certains travaux ont répondu à certaines questions relatives à notre travail :

1. Peut-on intégrer les interfaces tangibles dans la maison ?,
2. L'espoir mis sur les interfaces tangibles est-il bien fondé ?,
3. Les interfaces tangibles peuvent-elles servir à l'apprentissage ?

1.3.1 Peut-on intégrer les interfaces tangibles dans la maison ?

Cette section traite le fait qu'il est facile de proposer des interfaces tangibles pour la maison, qui ne sont pas forcément pertinentes.

Dans sa thèse, Ziraknejad propose des interfaces tangibles diffuses dans la maison [78]. Parmis les solutions qu'il propose, mentionnons :

1. Un siège tangible permet d'espionner d'autres pièces,

2. un jeu de plateau tangible et un affichage ambiant permet la surveillance de la consommation d'électricité et informe l'utilisateur,
3. un système d'éclairage automatique et un cadre multimédia servent à afficher des informations.

Tous les éléments proposés servent à faciliter l'accès à l'information, à configurer intelligemment l'environnement, à sécuriser, à être convivial, à être efficace énergiquement et à offrir de la communication. Parmi les points importants mis de l'avant dans ce mémoire, mentionnons que

- la maison déjà existante n'est pas conçue pour recevoir un système diffus ;
- les systèmes diffus ne sont pas compatibles entre eux, car trop spécifiques, des standards sont nécessaires ;
- les systèmes diffus devront être configurables par un usager et non un technicien ;
- les systèmes diffus devront être conçu pour un usage domestique ;
- il faut réfléchir à la dimension sociale.

Cependant, en réalité dans les projets présentés par Ziraknejad, il s'agit d'interfaces rendues tangibles, plutôt que de véritables interfaces tangibles. Le cadre multimédia n'est pas réellement une interface tangible, car il ressemble trop à un *PDA* muni un clavier simplifié. Je crois donc que les éléments présentés ne répondent pas à la question des interfaces tangibles telles que nous nous la posons. L'information n'est pas manipulée de manière tangible, seuls quelques boutons commandent l'affichage, le son ou la lumière.

Peut-on transformer en interfaces tangibles les objets usuels d'une maison ? En effet, comment transformer les portes d'un placards en interfaces tangibles ? L'interface tangible doit permettre de manipuler des données. La porte d'un placard peut d'une part représenter un variable dont l'ouverture et la fermeture modifient la valeur. Les portes d'un placard peuvent aussi être considérées comme permettant de donner ou interdire l'accès à de l'informations en conjonction avec d'autres interfaces tangibles.

Dans le cadre d'un assistant cognitif utilisant des interfaces tangibles au sein d'une maison, quelles sont les interfaces tangibles pertinentes ? Mon approche consiste à modéliser l'environnement et à manipuler ce modèle de la même manière que dans la réalité. Cette vision est inspirée du projet *Cognitive Cube* qui compare le modèle à obtenir et le modèle obtenu avec une fonction de similarité (cf. 1.4.2).

1.3.2 L'espoir mis sur les interfaces tangibles est-il bien fondé ?

Une étude a cherché à répondre à cette question en comparant des manipulations de fenêtres virtuelles et la manipulation de morceaux de papier [22]. Il a été demandé à plusieurs personnes de déplacer des fenêtres de manière habituelle, c'est-à-dire avec une souris, d'une part et de manière tangible, d'autre part. La fenêtre tangible correspond à un morceau de papier, mais on aurait pu aussi bien utiliser un écran tactile. Plusieurs fenêtres sont positionnées et la personne doit mémoriser leurs positions pendant une vingtaine de secondes. Enfin, la personne ferme les yeux et les fenêtres sont déplacées. La personne doit remettre les fenêtres à leurs positions initiales ou les placer en miroir. Les résultats montrent que la rapidité est meilleure avec le papier, donc avec la manière tangible, qu'avec la souris. Le nombre d'erreurs est aussi moindre pour le cas tangible.

Finalement, l'auteur affirme que les interfaces tangibles sont trop souvent abstraites et que si leur design était mieux abordé, il n'y aurait aucune raison de ne pas adopter les interfaces tangibles [22].

1.3.3 Les interfaces tangibles peuvent-elles servir à l'apprentissage ?

Plusieurs études indiquent que l'usage de systèmes d'assistance cognitive induisent une certaine forme d'apprentissage qui semble perdurer même lorsque les systèmes ne sont plus utilisés. Dans ce cas ne pourrait-on pas rectifier les interfaces tangibles d'assistance cognitive et d'apprentissage ? Cette section explore donc comment les interfaces tangibles ont été utilisée pour l'apprentissage et conclut que la recherche n'est pas encore suffisamment mature pour se prononcer sur ce sujet.

CTI [1] (Child Tangible Interaction) cherche à décrire comment les interfaces tangibles peuvent être utilisées pour développer l'intelligence des enfants entre quatre et douze ans. L'auteur développe cinq dimensions qu'il faut prendre en compte lors du développement d'une interface tangible dédiée à l'apprentissage :

1. actions dans l'espace,
2. perceptions,
3. comportements,
4. sémantique,
5. espace pour les amis.

Marshall [40] souligne que les discussions actuelles sur les interfaces tangibles sont très techniques, mais que peu d'études portent des effets sur les fonctions cognitives. L'auteur ne répond pas directement à la question, mais semble avoir la certitude que les interfaces tangibles sont intuitives. Elles favoriseraient l'exploration. Certains modèles peuvent favoriser l'apprentissage. Par contre, il met en avant le fait qu'il n'est pas évident d'unifier une structure (*framework*) pour toutes les applications.

Le projet *Smart Blocks* (cf. 1.3.6) propose une interface tangible pour l'apprentissage du calcul de surface et de volume pour les formes en trois dimensions. Il est donc possible d'utiliser les interfaces tangibles pour l'apprentissage.

1.3.4 Les tables interactives

Les tables interactives, aussi appelées surfaces interactives, appartiennent à un type d'interface tangible très répandue. Une table interactive est une table où les informations affichées dépendent des objets posés dessus. Les manipulations de ces objets vont interagir avec l'application qui en retour modifie l'environnement, par exemple l'affichage ou l'ambiance à travers le son et la lumière. La détection de la position des objets se fait généralement à l'aide d'une caméra vidéo qui voit l'ensemble de la scène. Enfin, une zone inactive permet de désactiver les objets qui ne sont pas nécessaires. Par exemple sur la figure 1.1, la zone blanche est la partie interactive de la table et l'affichage dépend des objets qui y sont déposés. En fonction de l'application, on pourra afficher des plans géographiques, des photos, etc.

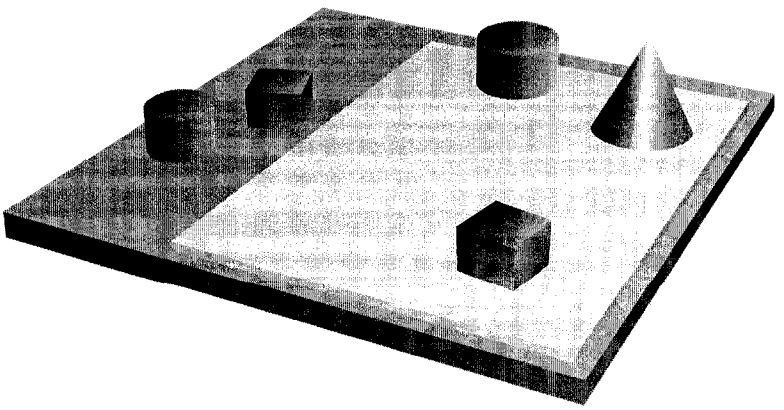

FIGURE 1.1 – Table interactive

Nous verrons dans cette section les systèmes qui sont classifiés en tant que table

interactive : Urp, Clay, reacTable, Sensetable, Planar Manipulator Display, Senseboard. Un habitat intelligent pourrait bénéficier de ce type d'interfaces tangibles, par exemple le bureau ou les plans de travail de la cuisine pourraient être des tables interactives. Dans la section 1.4, RecipeTable [27] et le travail de Lee [35] pour la conception d'une cuisine intelligente sont des exemples pertinents de l'utilisation de ce type d'interfaces tangibles.

Urp, Illuminating Light, Distributed Illuminating Light

Urp, Illuminating Light et Distributed Illuminating Light sont des projets qui ont été réalisés par la même équipe. *Urp* [70] permet l'analyse des vents et des ombres en fonction des positions de maquettes de bâtiments. Les ombres bougent comme si le soleil parcourait une journée et cela en fonction des saisons. Il est par conséquent possible d'interpréter les conséquences environnementales des architectures urbaines. La même équipe a produit un deuxième projet, Illuminating Light et Distributed Illuminating Light [69], pour faire de l'optique sans disposer de matériel comme les lentilles et supports. Il s'agit d'une simulation où la lentille est un objet que l'on place et la lumière résultante est projetée sur la table.

Clay

Le but du projet *Clay* [24] est de concevoir un paysage à l'aide d'une maquette ou d'un banc de sable. Des informations utiles sont projetées sur le paysage, comme la topographie. La détection se fait à l'aide d'un capteur infrarouge. Grâce à la détection des niveaux, le logiciel peut simuler les ombres suivant la période de la journée ou de l'année, comme dans Urp. Contrairement à *Urp*, il est possible de créer un paysage qui n'est plus plat. C'est d'ailleurs la seule table interactive qui n'est pas plate.

reacTable

reacTable [27] a pour but d'expérimenter la conception de musique électronique à l'aide d'une table interactive. Des blocs de formes élémentaires ont des fonctions différentes, par exemple par l'application de filtres reconnaissables sous la forme de carrés aux coins arrondis, ou de mixeurs reconnaissables sous la forme de pentagones. Des signaux passent de bloc en bloc et sont modifiés lors de leur passage dans un bloc. La rotation d'un bloc changera le paramètre associé à sa fonction. La projection des informations

sur la table permet de visualiser les signaux entre les blocs. Et finalement, le public peut entendre le résultat.

Sensetable

Sensetable [51] est composé d'une table interactive, où des informations sont projetées, d'objets localisables par ondes électromagnétiques avec une résistance variable et des deux écrans. La proposition est donc plutôt technique. En effet contrairement à la détection de position d'objets par vision, ce système n'est pas perturbé par la luminosité, ni par le champ de vision de la caméra. Les positions sont déterminées très précisément à l'aide de la technologie électromagnétique Wacom. Deux tablettes graphiques permettent d'augmenter la surface de détection. L'avantage est que le temps de latence est très faible, mais ce temps devient plus important lorsqu'un grand nombre d'objets bougent simultanément. L'utilisateur identifie un objet à l'aide d'une information projetée sur cet objet. Par contre, les flèches, qui représentent les liens entre les objets, sont projetées sur les objets et sur la table. L'épaisseur de l'objet provoque une cassure de la flèche, il devient alors difficile de suivre la flèche lorsque les objets sont proches entre eux.

Comme toute table interactive, le nombre d'applications est illimité. La technique utilisée est très précise et fiable, mais elle est coûteuse à cause de l'utilisation de deux tablettes graphiques de grande taille.

Audiopad [52] utilise la même technologie que *Sensetable*, mais de façon plus simple. La technologie utilisée est très bien expliquée dans l'article. *Audiopad* est une table interactive pour la composition musicale, comme *reacTable*.

Planar Manipulator Display

Planar Manipulator Display [59] propose une table interactive, qui utilise la détection de la position des objets à l'aide d'un signal infrarouge émis par deux diodes électroluminescentes positionnées sur chaque objet. Ainsi, la luminosité de la pièce ne pose plus de problème. L'identification d'un objet se fait à l'aide du signal émis par les diodes électroluminescentes. Les objets sont capables de tourner et d'avancer grâce à deux roues. Ils sont alimentés par des batteries, ce qui diffère de toutes les autres technologies vues auparavant.

Le projet permet à la fois un contrôle par l'humain et par la machine des objets physiques. L'application actuelle propose d'aménager une pièce où les objets représentent

les meubles. Les limites de la pièce sont projetées sur la table. Les configurations créées sont mémorisées. Ainsi, l'utilisateur peut rappeler une configuration et les meubles vont se remettre en place automatiquement. L'application doit donc gérer les collisions et planifier les déplacements des meubles.

Senseboard

Le projet *Senseboard* [26] augmente virtuellement un panneau pour organiser l'information et promeut le travail collaboratif. Une information est représentée par un petit bloc, et l'information associée est projetée sur le bloc. L'application permet les sauvegardes et les retours sur erreurs. Techniquement l'affichage offre une grille imprimée sur un tableau. Des lecteurs *RFID* (cf. 3.2) ont pour but d'identifier chaque élément posé sur la grille. De plus, l'écran est tactile et les objets s'attachent magnétiquement sur le tableau. *RFID* est utilisé, car il est fiable et rapide. L'application peut augmenter l'information lors de l'association d'une note et d'une bulle. La bulle ressemble à celle des bandes dessinées et correspond à une commande "voir les détails". Il y a plusieurs autres types de commande comme le groupage, l'exportation, etc.

Des expériences sous trois conditions ont été menées afin de déterminer l'efficacité du projet. Le premier cas de figure est le traitement de l'information avec du papier. Dans la seconde configuration, le Senseboard n'affiche pas l'information lorsqu'un objet est hors d'une contrainte représentée par une case. Dans le troisième cas, l'utilisateur se sert d'un stylo pour écrire l'information. Le dernier cas expérimenté correspond à l'utilisation du Senseboard avec toutes les fonctionnalités. Comme pour la plupart des projets sur les interfaces tangibles, les résultats sont très positifs lorsque toutes les fonctions sont disponibles. Par contre, les pires résultats sont obtenus lorsque l'interface Senseboard est très limitée. Le papier reste très performant avec des résultats très proches de ceux du *Senseboard* lui-même. Par contre, le papier ne présente aucun des avantages de sauvegarde et de transfert numérique tel que le *Senseboard* le propose.

Enfin le *Designer's Output* [32] ressemble au projet Sensetable. Il utilise un écran tactile *SmartBoard* et la vision pour photographier les notes écrites à la main. Il n'y a pas de couplage complet entre l'information écrite et la structure des notes, c'est-à-dire que les notes écrites restent des images, l'information n'est pas utilisable sans transcription humaine ou reconnaissance d'écriture, ce qui s'approche d'un *Senseboard* limité.

Actuated Workbench

Actuated Workbench [48] offre la possibilité d'utiliser une matrice de champ magnétique afin que les systèmes puissent déplacer des objets à sa guise. La technologie est donc différente de celle utilisée dans *Planar Manipulator Display.* Le résultat semble moins intéressant, car il ne propose qu'un seul objet et le positionnement se fait dans une matrice. Afin d'atteindre plus de positions, une technique de lissage est utilisée. Nous avons pensé réaliser le premier prototype avec un moyen similaire, jusqu'au moment où nous avons découvert *reacTIVision*, un framework libre avec le code ouvert et utilisant une simple webcam 4.3.

1.3.5 Token And Constraint (TAC)

Le Token+Constraint (*TAC*) [62] [68] joue avec les objets qui se placent dans des emplacements physiquement possibles. Les interactions sont directement liées avec les manipulations physiquement possibles des objets dans des emplacements virtuels en réels appelés contraintes (fig. 1.2). On distingue donc deux types de manipulations : l'association, le fait de poser ou de retirer un objet (*token*) d'un emplacement (*constraint*) et la manipulation de l'objet dans l'emplacement, comme le présente la figure 1.3.

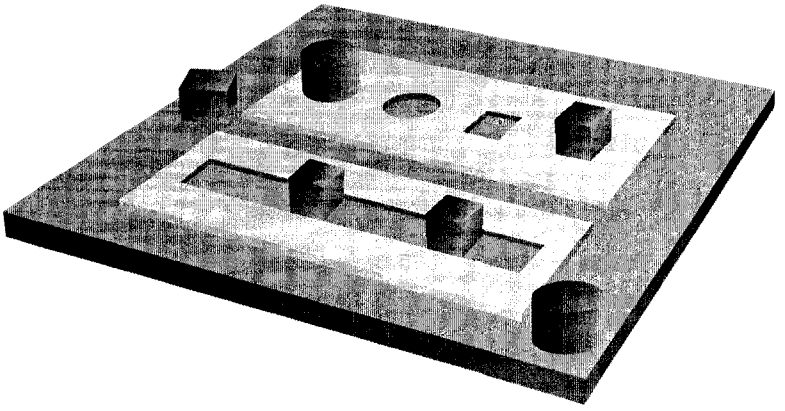

FIGURE 1.2 – Token and Constraint : positionnement dans les contraintes

Une partie de mon projet a consisté à donner une représentation et implémentation du TAC en langage orienté objets (chap. 4.1). Le TAC sera donc décrit plus en détail dans le chapitre 4. Cette section dresse la liste des projets qui utilisent des interactions de type *Token And Constraints* : Answering Machine et Mediablocks. Il est intéressant de lire cette section car il explique comment fonctionnent les interfaces tangibles réalisées

et qui répondent à la classification du modèle TAC, formalisé après la réalisation de ces projets. Nos prototypes ont été conçus sur ce modèle, il est donc important de lister les projets existants.

Answering machine

Answering machine [62] est un répondeur conceptuel qui représente les messages sous forme de billes. Suivant la présence ou non de billes (jeton) sur le répondeur (zone de stockage qui est une contrainte), la personne interprète facilement le nombre de messages reçus. Pour écouter un message, l'utilisateur saisit une bille (jeton), donc un message, et la place dans un emplacement de lecture (contrainte). L'utilisateur remet la bille dans le répondeur pour supprimer le message. Le concept montre bien ce qu'on peut faire en interface tangible, mais la réalisation n'est pas facile. Il faut un mécanisme pour sortir une bille du magasin du répondeur. La capacité de stockage du répondeur dépend du nombre de billes, alors qu'en informatique la capacité de stockage est infinie.

Mediablocks

Mediablocks est un système composé de multiples blocs qui stockent du contenu multimédia, au niveau conceptuel. Le système identifie chaque bloc (jeton) et retrouve les données qui lui sont associées. Les différents emplacements (contraintes) permettent la manipulation des données, comme le déplacement, l'affichage et l'impression. Les emplacements sont répartis dans des systèmes qui sont mis en réseau afin de récupérer les données, par exemple l'imprimante, le tableau de présentation et le système de manipulation des données. Avec Mediablocks, il est possible de préparer des présentations multimédias sans souris et clavier, simplement à l'aide d'une représentation physique des données virtuelles. Par contre, la métaphore nous semble incomplète. Cela ressemble à des clés de stockage *USB* qui possède du contenu multimédia. Tangible Video Editor (cf. 1.3.6) répond mieux à la problématique de la préparation de contenu multimédia.

1.3.6 Association d'objets

En terme d'interface tangible, l'association d'objets correspond à un jeu de construction où l'information est modélisée, enregistrée et modifiée à l'image de la forme construite physiquement. Dans la figure 1.4, le bloc en gris foncé ne participe pas à l'assemblage

contrairement à ceux qui sont clairs. En effet, le bloc foncé n'est pas connecté avec d'autres blocs qui sont connectés au système.

C'est la dernière catégorie d'interface tangible et bien que nous ne l'avons pas utilisée, il est important de lister certains projets intéressants qui pourront donner des idées pour de futurs travaux. De plus, Cognitive Cube 1.4.2, basé sur Active Cube, est un projet d'association d'objets qui mesure les performances des fonctions cognitives de la personne.

Topobo

Topobo [56] est un robot que l'on construit et que l'on manipule comme une poupée dans un premier temps. Puis dans un deuxième temps, les mouvements exécutés sur le robot sont reproduits grâce à des moteurs inclus dans les modules assemblés. Il y a une détection des actions faites lors de la manipulation par l'homme, puis une reproduction de ces actions par le système de manière autonome. Le projet *Super Cilia* (cf. 1.3.8) a repris ce concept de reproduction de l'action humaine par le système.

Active Cube

Active Cube [74] est un projet d'assemblage de cubes. La construction est représentée sous forme d'un arbre. La racine de cet arbre est déterminée grâce à un unique cube relié à l'ordinateur. Par contre, il est possible que deux blocs représentant des feuilles de l'arbre se rejoignent. Mais dans ce cas, l'arbre ne se referme pas. Car, comme dans tout arbre, un nœud ne retrouve pas à deux endroits différents, contrairement à un composite. Un algorithme est appliqué pour recomposer la structure physique. Aussi, si un cube est détaché alors qu'il avait des fils appartenant à cette boucle, les fils se déplacent sur l'autre branche de la boucle. Un détail contestable à mon goût car cela montre que la structure choisie pour représenter la forme n'est pas bonne. Sinon ce projet offre de nombreuses possibilités que le projet *Cognitive Cube* exploite par exemple.

Il y a un protocole de communication entre les blocs et un microprocesseur dans chaque bloc. Ce projet est un bon exemple d'informatique diffuse, ce que le laboratoire DOMUS veut construire à terme. La collaboration entre les blocs offre des possibilités intéressantes, notamment en matière d'intelligence artificielle et la programmation agent.

Des blocs ont parfois des fonctionnalités comme un gyroscope, un haut-parleur, un micro, un afficheur à *LED*, etc. Cela permet de construire une maquette d'avion qui commande un simulateur de vol, tel que le projet est présenté dans l'article. On ajoute

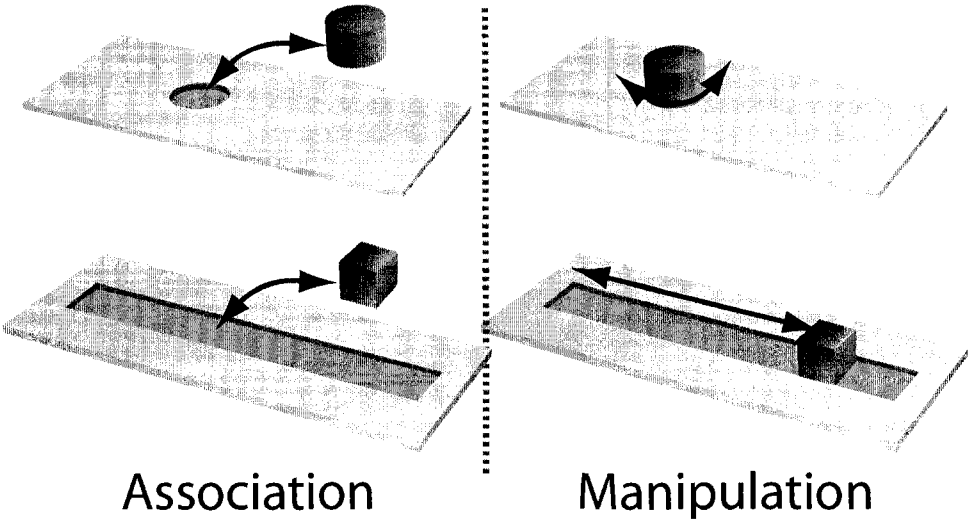

FIGURE 1.3 – Deux principes dans TAC : l'association et la manipulation

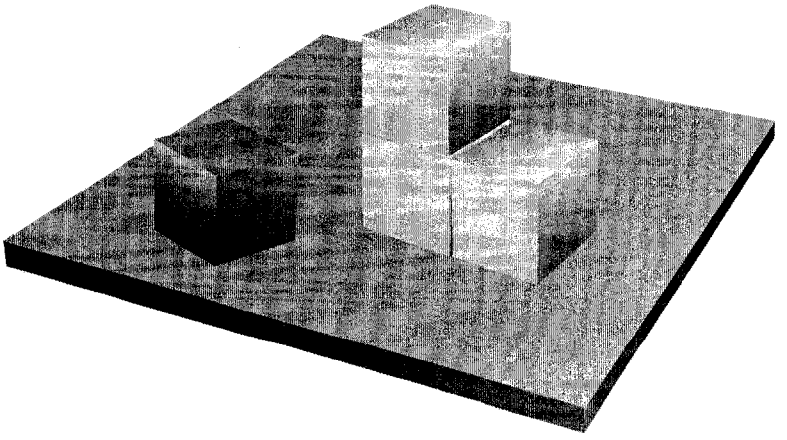

FIGURE 1.4 – Association et construction

ainsi des fonctionnalités à la structure.

Smart Blocks

Le but du projet Smart Blocks [15] est de proposer une interface tangible pour l'apprentissage pour les enfants. Les objets sont utilisés par les professeurs pour représenter des concepts. Afin d'apprendre à l'enfant à calculer la surface et le volume d'un objet en trois dimensions, Smart Blocks rend interactif ce calcul. Il y a deux modes :

- le mode exploratoire où l'enfant assemble des blocs et calcule la surface et le volume,
- le mode exercice où l'enfant répond à la question en construisant une forme adéquate.

Il est demandé de donner la surface et le volume d'une forme donnée ou de reproduire la forme qui a la surface et le volume donnés.

La réalisation se base sur *TUIML*, qui utilise le modèle *TAC*. Les questions sont équipées de puces *RFID* afin que le système puisse identifier la question posée. Pour calculer la surface, le système tient compte du nombre de cubes et du nombre de surfaces reliées. En effet, le volume dépend du nombre de cubes, alors que la surface tient compte du nombre de cubes et du nombre de faces reliées à d'autre cubes, par conséquent ces surfaces sont cachées. *RFID* est utilisé pour identifier les cubes et les connecteurs.

Tangible Video Editor

Le but du projet *Tangible Video Editor* [77] est de rendre tangible l'édition d'une séquence vidéo en associant les morceaux de vidéos avec ou non une transition appliquée entre deux vidéos. Chaque morceau de vidéo est stocké et lu dans un *PDA*. L'utilisateur assemble les *PDA* entre eux comme un puzzle. Il peut assigner des transitions spécifiques entre les vidéos collées. Enfin, le premier morceau assemblé est un bouton de lecture et donne le point de départ de la lecture de la séquence composée.

Le partage audio

Ce projet original propose la manipulation des droits de partage de musique à l'aide de blocs qui représentent des utilisateurs distants [13]. Suivant la position du bloc, debout ou couché, l'utilisateur à distance pourra écouter ou non de la musique. Les blocs s'associent entre eux pour former des groupes d'écoute et de partage. Enfin, l'intensité d'un voyant de couleur pour chaque bloc indique si l'utilisateur a écouté de la musique très récemment

(intensité forte), il y a longtemps (intensité faible) ou il y a encore plus longtemps (voyant éteint).

Ce projet a rendu très abstraite la manipulation des droits de diffusion en réseau de la musique pour créer une radio sociale. Les voyants lumineux permettent la notification abstraite et ambiante de l'état du système. Une métaphore repose sur la position des blocs permet de commander le système. Mais qui manipule les blocs ? Ils sont tous au même endroit !

1.3.7 Le papier interactif

Le papier interactif est constitué de papier amélioré par l'intégration d'une puce capable de fournir de l'information [53]. Les applications sont innombrables. Pour l'assistance cognitive, une imprimante pourrait écrire une tâche ou un rendez-vous sur des étiquettes. Si la personne veut de l'aide sur cette information, elle place le morceau de papier sur une borne spécifique. Elle peut obtenir par exemple le trajet qu'elle doit parcourir pour aller à son rendez-vous.

Hewlett Packard *RFID Memory Spot* est un projet commercial qui propose des tags *RFID* qui contiennent de l'information. Cette information est lue au passage d'un lecteur. Cela permet d'ajouter de l'information sur un document, de reprendre numériquement les informations essentielles ou de rendre hypertexte du papier physique.

1.3.8 Mélange des genres

Un grand nombre de projets répondent à leurs problématiques en associant différents types d'interfaces. Ces projets entrent dans le domaine des interfaces tangibles, car ils utilisent des objets physiques pour interagir avec le système.

Datatiles

Datatiles [57] est un projet où on place des plaques transparentes gravées. Lorsqu'une plaque est placée sur un emplacement de l'écran, une information est affichée en dessous. On manipule l'information à l'aide d'un stylet. L'écran est une tablette graphique. Ces interactions constituent la partie *TAC* de ce projet. Par ailleurs, il est possible d'associer les plaques entre elles afin de rajouter des interactions. Cette possibilité constitue la partie association d'objets du projet. Le stylet a des libertés de mouvement limitées à la forme gravée dans la plaque.

Enfin, citons le projet de Waldner qui ressemble à ce que *Datatiles* propose [73]. Ce projet reprend le concept de plaque transparente de *DataTiles* pour manipuler de l'information, notamment des images projetées sur une table interactive.

Conception et manipulation 3D

Les interfaces tangibles apportent aussi une nouvelle façon de manipuler des formes en trois dimensions. Aussi, réaliser des opérations sur des formes en 3D est souvent complexe. Par exemple, réaliser un plan de coupe à l'aide d'un clavier et d'une souris prend beaucoup de temps alors que les projets présentés ensuite, celui de Lee et de Doll, offrent des interactions immédiates.

Le projet de Lee [10] sert à concevoir une forme en trois dimensions. En effet, il est proposé un plan de coupe de la forme 3D à concevoir et l'utilisateur saisit le niveau de la coupe à l'aide d'un jeton. Ainsi de plan en plan, la forme en trois dimensions désirée prend forme. Cette forme est affichée sur un deuxième écran, ainsi que le plan de coupe afin de faire un contrôle en temps réel des actions effectuées. Le plan de coupe est une fenêtre en verre semi-transparent où est projetée la coupe de la figure en trois dimensions. Une caméra *CCD* analyse la présence du jeton et les manipulations faites lors de sa présence sur le plan de coupe.

Le projet de Qi [54] utilise la technologie pour la capture de mouvement en trois dimensions à l'aide de points de références. Trois objets servent à faire l'ensemble des manipulations : un cube représentant la forme manipulée, un cadre représentant le plan de coupe et une tige représentant un pointeur. Suivant la position relative entre ces trois objets, l'utilisateur manipule l'objet en trois dimensions.

La métaphore du plan de coupe semble avoir été bien maîtrisée par ces projets. *Doll* propose aussi cette métaphore en proposant de visualiser une radio du crâne. Une poupée et un plan permettent de régler la position du plan de coupe.

Media Control

Le projet *Media Control* [7] présente les fonctions multimédias (lecture, pause, suivant, etc.) sous forme de tiges, de pétales ou d'un cube. Dans le cas de la tige, il faut la saisir et la tordre pour activer la commande associée. La tige a une forme qui représente la commande, comme le carré pour l'arrêt, les deux barres pour la pause, etc.

Dans le cas de la fleur où l'utilisateur manipule des pétales pour activer la lecture,

mettre en pause, avancer ou reculer. Le pétale du haut sert à activer la lecture ou la pause si le pétale est à moitié ouvert. Le pétale de droite sert à sélectionner la piste suivante ou à avancer si le pétale est à moitié ouvert. Le pétale de gauche ressemble à celui de droite, mais pour reculer. Enfin, le pétale du bas sert à régler le volume. Le volume est silencieux lorsque le pétale est fermé et au maximum lorsqu'il est complètement ouvert.

Enfin, l'article propose un moyen de contrôle plus simple où les fonctions sont situées sur des faces d'un cube.

Cube

Le cube est souvent proposé comme interface d'entrée. Le projet *Cube* [33] propose un affichage sur chaque face en plus de commander l'ordinateur suivant la face vue par l'utilisateur. Trois capteurs de niveau sont orientés dans trois directions et donnent trois signaux faibles ou forts suivant la face choisie. Il est donc possible de déterminer quelle face est en haut, donc vu par l'utilisateur.

Cubik

Cubik [36] est un projet qui permet la manipulation de forme en trois dimensions à l'aide de trois résistances variables rotatives placées sur trois faces. Le problème est que l'interaction physique sur l'objet virtuel n'est pas la même. En effet, le cube physique ne suit pas la rotation du cube virtuel, manipulé par les commandes rotatives. Il faut donc manipuler les commandes rotatives au lieu du cube. Voici un projet qui offre une mauvaise métaphore.

Phidget

Les *Phidgets*[4] [16] sont un ensemble de petits modules *USB* ayant la même fonction que les widgets. En effet, l'objectif est de rendre sous une forme physique les *widgets* virtuels. On assemble les phidgets afin de concevoir l'interface physique souhaitée. L'*USB* est utilisé pour créer un arbre de Phidget, mais je ne retrouve pas la structure des modules USB nulle part dans leur librairie. La librairie fournie n'utilise pas les mêmes classes que *Swing* ou *AWT*. Le développeur doit donc reconstruire le modèle *MVC*, s'il le souhaite. Les Phidgets et la librairie permettent de réaliser rapidement un prototype fonctionnel avec du matériel programmable en Java.

[4]Physical widgets

L'analogie entre les objets physiques et les widgets se restreint uniquement à ce qui existe dans les interfaces graphiques actuelles et est un peu limitée, d'autant plus que les widgets se sont inspirés des objets physiques.

Rubber shark

Le projet Rubber Shark [9] permet la possibilité d'utiliser le poids des objets pour caractériser un objet. Cela pose la question du poids unique. Est-ce qu'il est possible d'identifier un objet par son poids. Le projet a classifié un certain nombre d'objets par leurs poids. Il est montré qu'il n'est pas possible d'identifier exactement un objet, car pour un même poids, il existe plusieurs objets. Par contre, l'application proposée peut utiliser cette possibilité. Le système jouera un son ou ouvrira une page web lorsqu'un poids donné est atteint. Il n'est pas nécessaire que ce soit tel ou tel objets, il faut seulement atteindre un poids. Ainsi pour accéder à un site déterminé, l'utilisateur devra poser un objet pour atteindre le poids souhaité. Une alternative à cette approche serait d'utiliser la pression du pied ou du bras.

Super Cilia Skin

Super Cilia Skin [55] imite le vent dans le blé à l'aide de multiples tiges que le système oriente, ce qui modifie la texture de la surface. Avec un grand nombre de tiges, le système est capable d'afficher une image monochrome. Les exemples donnés sont une application sur le sol ou sur les murs extérieurs d'un bâtiment. Cela va encore plus loin, car cette interface peut être manipulée par l'homme et les actions sont reproduites par le système, comme pour Topobo. Il est possible que dans un futur proche, les murs d'un bâtiment s'animent pour indiquer le sens du vent ou fournir de l'information contextuelle par exemple.

1.4 Les projets déjà existants liant l'assistance cognitive, habitat intelligent et interfaces émergentes

1.4.1 Habitat intelligent

Les grandes entreprises comme Intel [11] travaillent beaucoup sur le soutien à domicile des personnes âgées. Leurs travaux sont orientés vers la télévigilance et l'assistance [11].

L'assistance, par exemple pour la préparation d'une tasse de thé, repose sur la détection d'activité à partir de la localisation de la personne et des objets. La localisation est construite par inférence d'événements de capteurs ou directement avec l'utilisation de la technologie *RFID* ou infrarouge. La télévigilance prend une entre autres la forme d'un système de détection de chutes.

Intel propose les mêmes concepts que DOMUS. L'ajout des interfaces tangibles dans l'habitat intelligent pour l'assistance cognitive, peut apporter une innovation importante pour DOMUS.

1.4.2 Cognitive cube

Cognitive Cube [63] est un projet intéressant, car il montre une utilisation plus concrète d'Active Cube. Une construction est affichée sur un écran. Les cubes de cette construction possèdent une orientation indiquée à l'aide de bandes bleues sur certaines arêtes. La construction est représentée en trois dimensions et la représentation peut pivoter sur plusieurs axes. Il est demandé à l'utilisateur de reproduire physiquement cette construction. Le système mesurera le temps et la précision des actions. Il pourra donc établir un profil cognitif de la personne. On évalue ainsi les facultés de planification, la précision et la vitesse. L'expérience se termine lorsque la personne le souhaite, c'est-à-dire lorsqu'elle estime avoir mené à bien l'exercice. Le programme évalue la construction finale à l'aide d'une fonction de similarité, qui détermine si la forme est très proche ou non du but.

1.4.3 RecipeTable

RecipeTable[5] est une table interactive basée sur ReacTIVision qui propose des recettes de cuisine en fonction des ingrédients posés sur la table. Une interface tactile permet de sélectionner une recette dans la liste proposée. L'emploi de ce genre d'interface au sein du laboratoire DOMUS pourrait supporter la prise de décision sur le choix de recettes de cuisine par exemple.

[5] http ://www.recipetable.net/

1.4.4 Intégration d'interfaces émergentes dans une cuisine

Lee [35] a travaillé sur l'intégration des interfaces émergentes dans la cuisine. Il utilise des projections d'informations sur les placards, les plans de travail et le réfrigérateur. Cette approche est similaire à la réalité augmentée [61]. Une interface procédurale assiste la préparation de recettes de cuisine. Cette interface est affichée sur le plan de travail et est tactile. Les mouvements sont détectés à l'aide de systèmes de vision par ordinateur. Toutefois, la cuisine devient très complexe, peut-être trop pour la personne atteinte de troubles cognitifs.

1.4.5 SentientArtefacts

Le projet *Sentient Artefacts* [31] propose propose l'"augmentation" des objets de la vie quotidienne, comme le miroir, la brosse à dents, les tasses, les chaises, les lampes et la table. Pour le laboratoire DOMUS, ce genre d'analyse et de construction s'avèrent sûrement prometteuses. Toutefois il est fort probable que des solutions commerciales émergent entre-temps, comme des interfaces ambiantes à la manière de lampes de présence. Il s'agit d'un domaine très actif en ce moment.

Chapitre 2

L'assistance cognitive à l'aide d'interfaces tangibles

Ce chapitre explique en quoi consiste notre travail. La section 2.1 explique comment nous nous sommes approchés de la réalisation d'un assistant cognitif pour la préparation de repas. Le premier prototype virtuel est décrit dans la section 2.2 et le second prototype réel est quant à lui décrit dans la section 2.3. Les choix techniques et les choix des modèles, en réponse aux besoins des deux prototypes, sont énumérés dans la section 2.4. La portée de notre travail est détaillée dans la section 2.5.

2.1 Vers un assistant cognitif pour la préparation des repas

La réalisation s'articule avec deux applications autour d'un même scénario : la préparation d'un plat de riz.

Le premier prototype a pour but de virtualiser les manipulations faites lors de la préparation d'une recette de cuisine. Il permet de tester rapidement la faisabilité et de montrer les mécaniques internes du programme, sans avoir des liens avec une cuisine réelle.

Le deuxième prototype reprend la préparation d'un plat de riz mais cette fois dans une cuisine réelle. Il permet de montrer que l'architecture est effectivement réalisable en grandeur nature.

Les deux prototypes partagent beaucoup d'aspects et permettront la construction

d'un cadre structurel pour de futures applications.

2.2 Une cuisine virtuelle tangible

Le premier prototype repose sur la mise en place d'une assistance virtuelle tangible. Les objets de la vie réelle sont symbolisés par des blocs posés sur une table interactive. Ces objets virtuels correspondent à des contenants, tels que des casseroles, ou des ingrédients, comme de l'eau, du riz. Des contraintes sont affichées sur l'écran représentant les actions. Les actions virtuelles possibles sont les pendants de celles de la vie réelle : verser, cuire, couper, mélanger et d'égoutter du riz dans une passoire.

La vision permet une grande précision, une grande flexibilité et une grande facilité à mettre en place. La table interactive sera donc conçue grâce à reacTIVision (4.3), qui offre une plateforme complète, performante et ouverte.

2.3 Les interfaces tangibles dans une cuisine réelle

Le second prototype réalise un assistant cognitif à l'aide d'interfaces tangibles dans une cuisine réelle. L'application utilise les vrais objets. L'identification d'objets à l'aide de RFID sera utilisée pour construire des fonctionnalités similaires au cas virtuel.

Toutefois, des nuances importantes sont apportées. Les actions virtuelles sont traduites en termes de déplacements et d'utilisation d'objets réels. Par exemple dans le cas virtuel, l'action de passer le riz cuit se fait en posant l'objet sur la contrainte "passer". La même action, dans le cas réel, est détectée par la satisfaction des conditions suivantes : la passoire en tant que jeton est d'abord posée dans la contrainte évier ; puis la casserole passe au-dessus de la passoire, prise en tant que contrainte.

La modélisation TAC diffère donc, mais l'architecture de l'application reste identique. Seuls les modules concernant les aspects techniques liés à la détection des objets sont différents.

2.4 Choix techniques et choix de modèles

Notre objectif est d'intégrer les interfaces tangibles et l'assistance cognitive. D'après les trois grands types d'interfaces tangibles vues dans le chapitre 1 (table interactive, *token+constraint* et assemblage), le type *token+constraint* semble le plus approprié pour

ce projet. Ce modèle permet de modéliser les interactions entre des objets physiques et le monde virtuel. Il existe une adéquation parfaite entre le modèle *TAC* et le suivi d'activités dans un espace physique non limité. Par exemple, le modèle *TAC* permet de modéliser facilement une assiette qui sort du placard et qui entre dans l'évier. L'utilisation de cartes iconiques associées à des bornes d'interrogation et d'assistance, peut être facilement modélisé en *TAC*.

Une infrastructure reposant sur les tables interactives a été écartée pour les raisons suivantes :

1. La table interactive nécessite, comme son nom l'indique, une table améliorée capable d'afficher de l'information et détecter la position des objets qui y sont posés.
2. Une table interactive utilise une interface graphique. Nous avons vu à la section 1.2 que de telles interfaces sont peu adaptées pour les personnes ayant des déficits cognitifs.
3. Une table interactive ne s'intègre pas complètement et partout dans l'environnement. En général, seulement une ou deux tables et quelques plans de travail de la cuisine existent dans une maison.
4. Il est possible de modéliser les interactions de la table interactive à l'aide du modèle *Token+Constraint*. Par exemple, l'application virtuelle décrite à la section 2.2 a réalisé cette approche.

De son côté, l'approche fondée sur l'assemblage d'objets ne semble clairement pas convenir à la construction d'une interface tangible pour l'assistance cognitive. Bien que le projet *Cognitive Cube* ait réussi à appliquer ce modèle pour des tests de fonctions cognitives, nous ne voyons pas pour le moment comment un assemblage de Lego ou d'autres objets peut aider une personne à réaliser une recette, à s'habiller, etc.

Ainsi, notre travail utilisera le modèle *Token+Constraint*, qui apporte des réponses techniques pour la réalisation de notre prototype. En particulier, le *TAC* propose un modèle de l'environnement (les variables), qui sera à la base du raisonnement lié à la détection d'activité à des fins d'assistance cognitive.

2.5 De la portée de notre travail

Dans le cadre des problématiques et de la méthodologie du laboratoire DOMUS, des interfaces homme/machine, plus spécifiquement des interfaces tangibles, et de l'assistance

cognitive, notre travail a mis l'accent sur les aspects techniques suivants :

1. architecture modulaire,
2. configuration de l'architecture par des fichiers, ce qui permet la personnalisation complète de l'application,
3. réutilisabilité, grâce notamment aux patrons de conception et au langage orienté objet Java,
4. application du modèle *Token+Constraint*,
5. préparation à l'assistance cognitive,
6. suppression de la souris, du clavier et de l'écran,
7. respect des fonctionnalités.

Les aspects suivants vont au-delà de la portée de notre travail :

1. conception et validation au contact de la clientèle,
2. réalisation d'un système d'assistance cognitive intelligent et évolué.

Enfin, quelques interfaces ambiantes seront proposées. Elles n'entrent pas comme telles dans le domaine des interfaces ambiantes, mais elle complète l'application en permettant de supprimer l'écran.

Chapitre 3

Les aspects techniques

Ce chapitre traite des questions techniques qui se sont posées dans la construction des prototypes. Tout d'abord, le cadre de travail (*framework*) (3.1) est présenté. Un cadre de travail permet de structurer les programmes informatiques, de créer des librairies et des structures réutilisables, qui seront adoptées et personnalisées pour les différentes réalisations. Ensuite, OSGi répond aux besoins d'architectures modulaires et orientées services. Dans notre travail, seule la modularité est importante. La technologie RFID de Symbol est discutée dans la section 3.2. L'utilisation des Phidgets est présentée dans la section 3.3. Le système *reacTIVision* (3.4) est présenté sous sa forme originale, en conformité avec le site web du projet. Ensuite, la programmation modulaire et l'intégration continue sont présentées dans la section 3.5. Enfin, la plateforme *OSGi* (3.6) est introduite afin d'expliquer les enjeux et les possibilités offertes par cette plateforme.

3.1 Le cadre de travail (Framework)

Un *framework* est une structure conceptuelle de base, une collection de classes qui représentent des conceptions réutilisables pour construire des applications différentes. Un *framework* est composé en partie de classes abstraites et d'interfaces, et en partie d'implémentations de ces classes abstraites.

Voici les caractéristiques d'un bon *framework* :

Extensible Un *framework* possède une structure claire qui définit des classes abstraites et des interfaces qui seront implantées ou héritées. L'héritage permet de changer le comportement d'une classe en réimplantant certaines méthodes prévues pour cet usage.

Contrôle inversé Dans une librairie classique, le flux de contrôle est sous la responsabilité de l'application. Les classes de la librairie sont des esclaves de l'application, alors que dans un *framework*, certaines régissent le flux de contrôle de l'application.

Construction par patron de conception Les patrons de conception permettent de rendre réutilisable et flexible les structures de l'application. La structure utilise donc les patrons de conception pour sa mise en relation avec l'application et/ou d'autres structures. Les patrons de coopérations tels que la méthode type (*Template method*) et stratégie (*Strategy*) permettent à la structure d'être utilisée facilement.

Complet Toutes les classes de la structure sont unifiées pour un ensemble d'applications. Il faut généralement trois applications concrètes pour détecter les points communs et construire une structure à peu près complète.

Adaptable Les spécificités du *framework* par rapport à la plateforme doivent être isolées, par exemple une interface graphique prévue que pour Linux.

Efficace Le *framework* ne doit pas consommer trop de ressources CPU ou mémoire.

Sûr Le *framework* est fonctionnellement valable, c'est-à-dire qu'elle ne doit pas être perturbée par un comportement externe défectueux, par exemple une exception lancée en dehors de la structure même.

Simple La structure doit avoir une organisation claire et équilibrée.

Il est important de respecter tous ces points mentionnés afin de construire de bons *framework* qui seront réutilisés dans le futur. Notre travail consiste à créer un *framework* du modèle *TAC*. Le fait de construire deux prototypes permet de répondre au fait que le *framework TAC* sera complet. La réalisation et le test du *framework* permet de montrer que notre travail est sûr. *TAC*, de par sa définition, contrôle les entrées de l'application.

3.2 Symbol RFID

Radio Frequency Identification [65] [50] (*RFID*) est une technologie d'identification similaire aux systèmes à code-barres, mais sans-fil. *RFID* utilise les champs électromagnétiques pour alimenter des puces. Une puce transmettra alors son identité en utilisant

le champ électromagnétique. Suivant la fréquence de fonctionnement, le type de champ et les antennes utilisés, les comportements et les propriétés sont différents, par exemple la distance de détection.

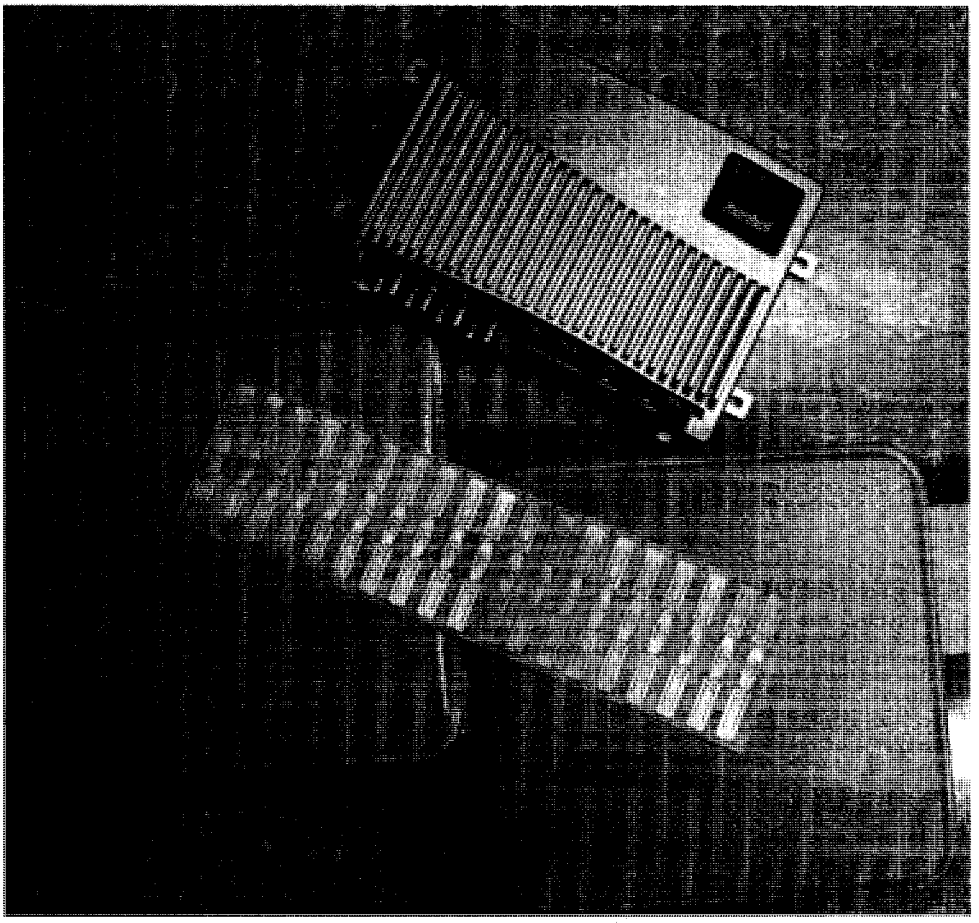

FIGURE 3.1 – Symbol RFID XR 400, deux antennes et des tags RFID

La technologie que nous avons utilisée est le *Symbol RFID XR 400* (fig. 3.1). Elle permet d'identifier à longue distance les tags (de quelques dizaines de centimètres à plusieurs dizaines de mètres selon la puissance choisie). On s'en sert dans la cuisine pour l'identification dans les placards, la cuisinière et le lavabo.

Un boîtier lecteur tel que le XR 400 peut recevoir quatre antennes, que nous pouvons disposer comme nous le voulons. Une antenne est en fait un couple d'antennes dont une antenne est émettrice, qui sert à alimenter les puces, et l'autre antenne est réceptrice, qui reçoit les signaux envoyés par les puces. Ces deux antennes, sur le plan technique, sont un seule antenne sur le plan logique, et elles sont souvent côte à côte. Ce boîtier contient

un système d'exploitation embarqué, comprenant un serveur web pour la configuration et la gestion des événements. Pour recevoir un événement, il y a trois possibilités : soit le lecteur *RFID* agit comme serveur et un client envoie des requêtes *HTTP* en *XML* de mise à jour aux quelles il répond en *HTTP* et *XML*, soit le lecteur *RFID* agit comme client qui envoie automatiquement à un serveur les événements en *HTTP* et *XML*, ou soit le lecteur *RFID* reçoit des paquets binaires et répond de façon binaire.

Les étiquettes RFID ont trois états : invisible, visible et inconnu. Les messages *XML* des requêtes sont les suivants : *login*, *queryEvents*, *queryTags*, *startPolling*, *getVersion* et *setDateTime* (fig. 3.2).

Un modèle *RFID* unifié et plusieurs implémentations *RFID* sont décrits dans la section 4.4.

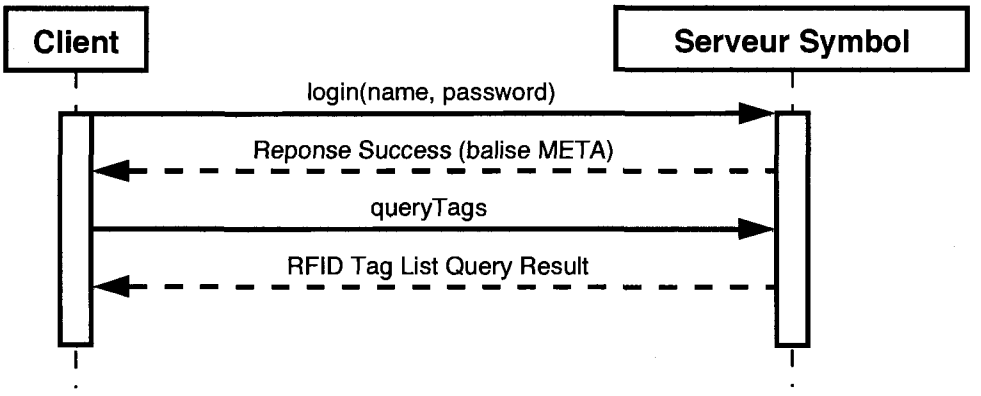

FIGURE 3.2 – Exemple de communication la prise de connaissances des tags RFID vus par les antennes

3.3 Phidget

Un Phidget est une carte d'acquisition et/ou de commande avec une interface USB 1.1 (Universal Serial Bus). Il existe plusieurs types de cartes pour différentes applications :

- balance électronique ;
- acquisition de données provenant de capteurs, tels qu'un accéléromètre, un capteur de luminosité, un capteur de pression, etc. ;
- commande de servo moteurs ;
- lecteur RFID ;
- etc.

L'utilisation d'un *Phidget* [16] est très simple grâce à la librairie en sa version 2.1. Cette librairie respecte les normes *Java*, ce qui n'était pas le cas auparavant. Le code est ouvert, mais pas l'architecture des modules physiques. Il existe une classe par élément Phidget ; les méthodes très explicites permettent une utilisation immédiate. La librairie *Phidget* en *Java* utilise *JNI*[1] pour se brancher à la librairie native installée sur l'ordinateur ou à un service web qui accepte les communications à distance.

Comme la librairie native a rencontré des problèmes lors de la mise à jour de *Mac OS X* et de *Linux*, et non *Windows*. Il est préférable d'utiliser les *Phidgets* sous *Windows*. Un forum est aussi mis à la disposition des clients afin de discuter des développements futurs et des problèmes techniques rencontrés.

Le fonctionnement d'un *Phidget* est simple. Il y a une phase d'ouverture avec les méthodes *openAny* et *open*. La méthode *open* prend le numéro de série du Phidget, qu'on ne connaît pas a priori. Pour cela, il faut ouvrir individuellement un module avec la méthode *openAny* et prendre le numéro de série à l'aide de la méthode *getSerialNumber*. Le service *Phidget* sous *Windows* peut fournir ces informations.

Une fois ouvert, le *Phidget* n'est pas encore attaché. Un *Phidget* est dit "attaché" lorsqu'il est branché et détecté par les pilotes systèmes *Phidget*. L'application peut soit attendre jusqu'à ce que le *Phidget* soit attaché avec la méthode *waitForAttachment*, ou ajouter un objet *AttachListener* avec la méthode *addAttachListener*. L'interface *AttachListener* définit une méthode qui est invoquée lorsque le *Phidget* est attaché. Le même procédé sert pour le détachement du *Phidget*. Pour fermer définitivement un *Phidget*, il faut appeler la méthode *close*. La figure 3.3 présente un exemple de code qui ouvre un *Phidget RFID* et qui affiche l'entrée et la sortie de tags *RFID*.

En fonction du module *Phidget* utilisé, les méthodes sont différentes. Le module est soit un ensemble de capteurs, auquel cas des écouteurs (*listeners*) peuvent être ajouté, ou un ensemble d'actionneurs, auquel cas un ensemble de méthodes est spécifié dans la classe spécifique au type du *Phidget*. Le site web *phidgets.com* contient toutes les informations nécessaires quant aux fonctions de chaque module.

La figure 3.4 présente un exemple d'assemblage de deux *Phidgets ServoMotor* qui servent à pointer des informations imprimées, à l'instar des compteurs de vitesse d'une voiture.

[1] *Java Native Interface*

```
import com.phidgets.PhidgetException;
import com.phidgets.RFIDPhidget;
import com.phidgets.event.TagGainEvent;
import com.phidgets.event.TagGainListener;
import com.phidgets.event.TagLossEvent;
import com.phidgets.event.TagLossListener;

public class PhidgetRfidTagReader implements
                TagGainListener, TagLossListener {

        private RFIDPhidget phidgetRFID = null;

        public PhidgetRfidTagReader() {
                phidgetRFID = new RFIDPhidget();
                phidgetRFID.addTagGainListener(this);
                phidgetRFID.addTagLossListener(this);
                phidgetRFID.openAny();
                phidgetRFID.waitForAttachment();
                phidgetRFID.setLEDOn(false);
                phidgetRFID.setAntennaOn(true);
        }

        public void tagGained(TagGainEvent ae) {
                String s = new String(ae.getValue());
                System.out.println("Tag " + s + " entre dans le lecteur RFID.");
        }

        public void tagLost(TagLossEvent ae) {
                String s = new String(ae.getValue());
                System.out.println("Tag " + s + " sort du lecteur RFID.");
        }
}
```

FIGURE 3.3 – Exemple d'utilisation de Phidget : cas du Phidget RFID

3.4 reacTIVision

reacTIVision [29] est un *framework* distribué pour construire rapidement une table interactive (fig. 3.5). La solution technique proposée par *reacTIVision* est souvent employée pour construire une table interactive : une caméra pour détecter la position des objets physique sur la table et un projecteur pour afficher des informations sur la table (section 1.3.4). Trois modules sont définis :

1. une application native multiplateforme écrite en *C++* ayant pour fonction de la détection de formes ; cette application correspond au module "reacTIVision" de la figure 3.5 ;
2. l'application développée utilise les informations collectées par l'application de détection de formes ; cette application correspond au module "TUI Application" de la figure 3.5 ;
3. un protocole (*TUIO*) de communication entre les deux applications qui utilisent *UDP* ; ce protocole de communication correspond au lien "TUIO" de la figure 3.5.

Une caméra *USB 2* 640x480 (*VGA*) est grandement suffisante pour un bureau. Il est possible d'utiliser d'autres caméras *USB2* ou *IEEE1394* (*FireWire*) ayant une plus grande résolution, mais l'image fournie ne doit pas être entrelacée, c'est-à-dire qu'il n'y ait pas de saut de ligne. La puissance de traitement requise est dépendante de la résolution et de la qualité de l'image. Il est conseillé d'avoir une bonne lentille et un éclairage satisfaisant afin d'améliorer la qualité de l'image. D'après les résultats fourni [29], les capteurs *CCD* sont plus performants que les capteurs *CMOS*. Une lentille avec un angle de vue large aidera à obtenir une surface de table importante, mais la mise au point peut ne pas être uniforme sur certains modèles. Pour adapter une lentille grand-angle, il faut disposer d'une caméra avec une lentille interchangeable. La mise au point de la caméra aura un impact sur la qualité de l'image, car la détection des objets ne sera pas bonne si l'image est floue. Les webcams ont une mise au point automatique. Il est aussi possible d'utiliser l'infrarouge pour l'éclairage car les capteurs *CCD* sont sensibles à cette lumière (s'il n'y a pas de filtre).

Si la caméra est placée de côté ou utilise des miroirs, l'image sera distordue. L'application native de détection contient toutes les opérations nécessaires pour corriger les problèmes de distorsions. Il faut calibrer le logiciel avec plusieurs images puis il se corrigera automatiquement en obtenant des images droites. La mise au point se réalise avec le projecteur et la caméra installés ensemble. Les images seront simplement projetées sur

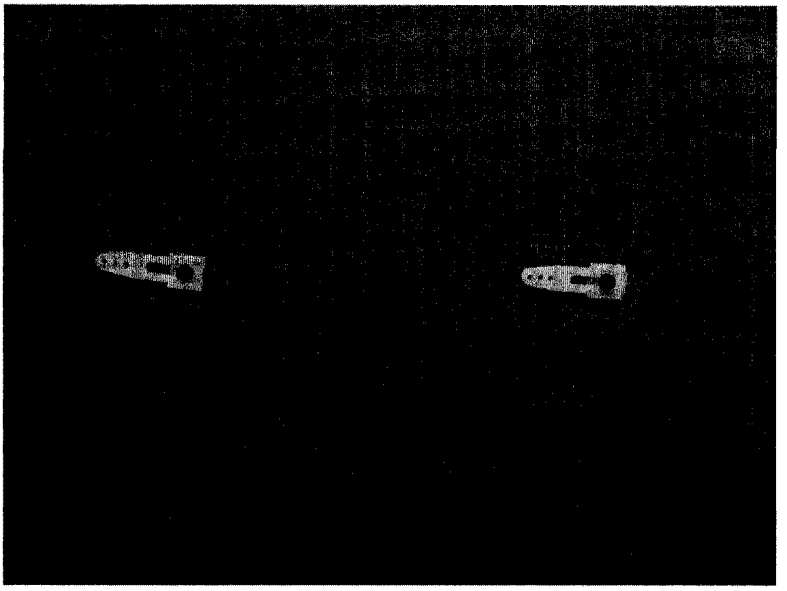

FIGURE 3.4 – Cadran ambiant utilisant deux Phidgets ServoMotor

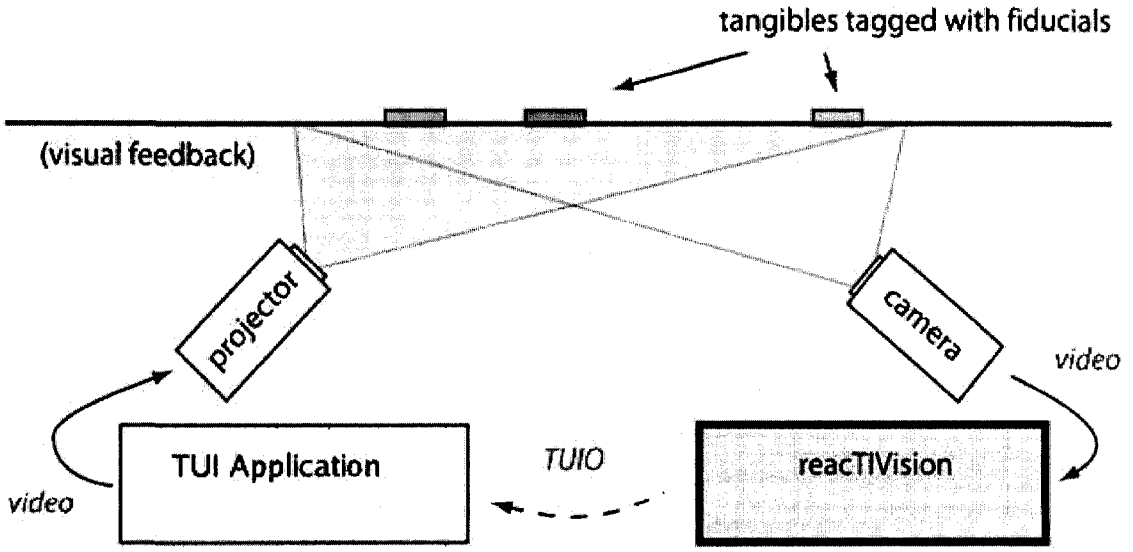

FIGURE 3.5 – Structure d'une table avec reacTIVision (source : [29])

la table.

Le logiciel détecte des formes imprimées et apposées sur des objets physiques, par exemple des cubes en bois ou en plastique. Les formes apposées sur les objets peuvent être différentes grâce à un mécanisme de choix du moteur de formes inclus dans le moteur reacTIVision. Par défaut les formes utilisées sont de type "*amoeba*", des formes qui ont été obtenues par algorithme génétique.

L'application qui utilise les informations de l'application de détection (reacTIVision) peut être programmée en n'importe quel langage, car les informations sont envoyées par *UDP*. Des exemples en *C++*, *C#* et *Java* sont fournis sur le site web de reacTIVision[2].

En particulier, l'exemple en *Java* représente la surface vue par la caméra et les objets positionnés. Au niveau interne, une classe *TuioObject* définit un objet détecté et une classe *TuioClient* qui met à jour la position des *TuioObjects*. L'affichage est ajouté au *TuioClient* en tant que *TuioListener*. C'est au développeur d'implanter son propre *TuioListener* qui recevra toutes les informations nécessaires. L'implémentation de *TuioListener* permet donc au développeur de créer sa propre application.

En outre, une application *Java* permet de simuler l'application de reconnaissance de formes (fig. 3.6). Il s'agit d'une interface graphique qui représente la table et un nombre d'objets que l'on peut déplacer virtuellement. Les informations sont transmises aussi par *UDP*. Cela permet de concevoir l'interface tangible sans réaliser la table physiquement et donc de développer et tester encore plus rapidement. On regrettera toutefois l'absence d'un enregistreur et d'un lecteur de scénario.

Nous avons utilisé des caméras web classiques : une *Logitech QuickMessenger* et une *Apple iSight*. Toutes les caméras reconnues par le système sont utilisables, car l'application *reacTIVision* utilise les pilotes systèmes. La table n'a pas été réalisée, toutefois les tests avec les caméras sont positifs. Aucune configuration spéciale n'a été faite lors de l'utilisation de *reacTIVision*, par exemple la calibration interne du logiciel.

3.5 Programmation modulaire

Le laboratoire DOMUS a mis en place une structure permettant un développement continu et modulaire en combinant *Ivy*, *Ant* et *CruiseControl*. *Ivy* permet la résolution de dépendances de manière automatique. *Ant* défini un langage de script de compilation en *XML*. *CruiseControl* est un système d'intégration continue qui compile les programmes

[2]http ://mtg.upf.edu/reactable/ ?software

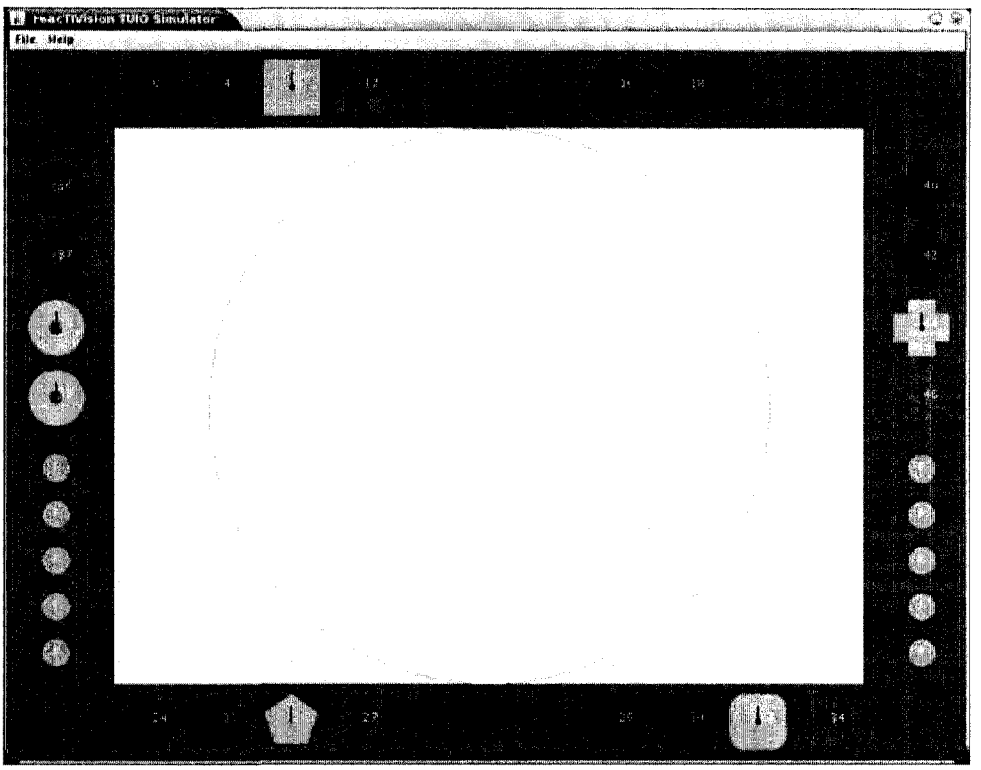

FIGURE 3.6 – Photo d'écran du simulateur reacTIVision (source : site web)

chaque jour ou dès qu'une modification au code est apportée. Après que le programme soit compilé, le résultat est publié sur l'intranet avec son numéro de version pour que le système de résolution des dépendances puisse télécharger la version adéquate de ce module, car il peut-être utilisé par d'autres modules avec des versions différentes. Enfin, les projets dépendants sont recompilés pour prendre en compte les changements du module. *CruiseControl* produit des rapports des compilations et lance les tests unitaires associés à chaque version compilée. Les différences entre les versions sont visibles sur chaque page de compilation. Il devient ainsi possible de retracer les modifications qui auraient pu casser toute la structure d'une application. La mise en place de ce système a donc permis de rendre modulaire toute l'infrastructure du DOMUS et de favoriser la réutilisabilité des modules construits par les étudiants.

Dans un tel contexte de développement continu et modulaire, la définition de concepts et de modèles fondamentaux est importante. Les modules deviennent probablement de plus en plus simples pour ne représenter qu'un seul concept. Par contre, la gestion des dépendances n'en sera pas plus facile. Notre travail a profité pleinement de ces possibilités. Vingt-et-un modules ont été définis dans notre travail. Chaque module représente un concept qui pourrait être utilisé dans d'autres applications. Les modules respectent la définition de *framework*. Certains modules isolent des spécificités matérielles comme le *Symbol RFID* ou les *Phidgets*.

3.6 OSGi

La structure d'intégration continue du laboratoire DOMUS (section 3.5) permet de développer des applications modulaires d'une façon rigoureuse. *OSGi* est une plateforme modulaire orientée services. La plateforme *OSGi* permet d'assembler les modules entre eux et de développer des applications orientées services. Des fonctionnalités d'autodéploiement résolvent les dépendances entre les modules et chargent automatiquement les modules nécessaires. C'est pourquoi mon travail est compatible avec *OSGi*. Mon projet comporte de nombreux modules avec un nombre important de dépendances. Pour qu'il soit utilisable immédiatement par les projets futurs, la préparation à *OSGi* est importante. J'expose donc les éléments nécessaires pour développer une application sur *OSGi*. Toutefois, notre application n'est pas orientée services en tant que telle. La plateforme *OSGi*, qui a été utilisée, est *Knopflerfish* version *R4*[3]. DOMUS peut donc tirer de nom-

[3]http ://www.knopflerfish.org/

breux avantages de cette architecture. DOMUS a donc choisi de construire les futurs projets sur cette plateforme. Dans *OSGi*, on appelle un module "*Bundle*". Pour qu'un bundle puisse être exécuté, un *BundleActivator* doit être implanté et déclaré dans le fichier *Manifest* (fig. 3.7). Le fichier *Manifest* déclare toutes les dépendances du *bundle.*

```
Manifest-Version: 1.0
Bundle-Name: kitchenaid-simulated
Bundle-SymbolicName: kitchenaid-simulated
Bundle-Version: 1.0.0
Import-Package: ca.usherbrooke.domus.ihm.tui.tac, ca.usherbrooke.domus
 .ihm.tui.tac.model, ca.usherbrooke.domus.kitchenaid.framework, ca.ush
 erbrooke.domus.kitchenaid.framework.assistance, ca.usherbrooke.domus.
 petri.xml, org.osgi.framework
Bundle-Activator: ca.usherbrooke.domus.kitchenaid.KitchenAidMain
Export-Package: ca.usherbrooke.domus.kitchenaid, configuration.kitchen
 aid.petri, configuration.kitchenaid.tac
```

FIGURE 3.7 – Le fichier Manifest du Bundle OSGi kitchenaid-simulated

L'interface *BundleActivator* oblige la classe qui sera lancée par la plateforme OSGi, à implanter les méthodes suivantes :

- public void start(BundleContext context) throws Exception
- public void stop(BundleContext context) throws Exception

Le *BundleContext* peut être ignoré pour un usage simple, bien qu'il serve à établir un contact avec la plateforme et les autres *Bundles.* Elle permet par exemple de récupérer l'instance de son *Bundle* en appelant *context.getBundle().* Cela peut être utile si le *Bundle* lui-même veut se fermer ou pour changer des propriétés.

La génération d'un *Bundle* est facilitée par une extension d'*Eclipse* fournie par la plateforme *OSGi Knopflerfish R4.* Cette extension génère le fichier *Manifest* et complète les packages importés et exportés. Le travail est très simplifié. Pour que le *Bundle* soit exécutable, il faut implémenter un *BundleActivator.*

Chapitre 4

Une architecture pour intégrer les interfaces tangibles à l'assistance cognitive

En réponse aux problèmes posés dans l'introduction et au chapitre 2, les modèles proposés sont présentés dans ce chapitre. Tout d'abord, j'ai choisi de réaliser les interfaces tangibles pour la cuisine. En effet, la cuisine possède de multiples objets, qui sont manipulés afin de réaliser une recette de cuisine. Suivant les actions faites par l'utilisateur, un assistant validera ou non ces actions en modifiant des affichages ambiants, comme des lampes, des horloges ou des sons. L'assistance cognitive dépendra des informations provenant des interfaces tangibles.

L'architecture du programme est divisée en trois couches (fig. 4.1). La couche intermédiaire comprend le modèle *TAC* (section 4.1) et le réseau de Petri (section 4.2). Les couches matérielles proposées sont composées de la détection par vision à l'aide de *reacTIVision* (section 4.3), du *RFID* (section 4.4) et des capteurs diffus (section 4.5) déjà existants dans le laboratoire. Les couches supérieures traitent de l'affichage d'informations, de la configuration, des modèles de réalisation des recettes et de l'assistance. Pour décrire les applications et comment les couches sont liées entre elles, nous procédons en trois sections :

- les points communs entre les deux applications, afin de définir l'articulation du *TAC*, des réseaux de Petri et des affichages ambiants (section 4.7),
- les liaisons et la configuration pour l'application virtuelle (section 4.8),
- les liaisons et la configuration pour l'application réelle (section 4.9).

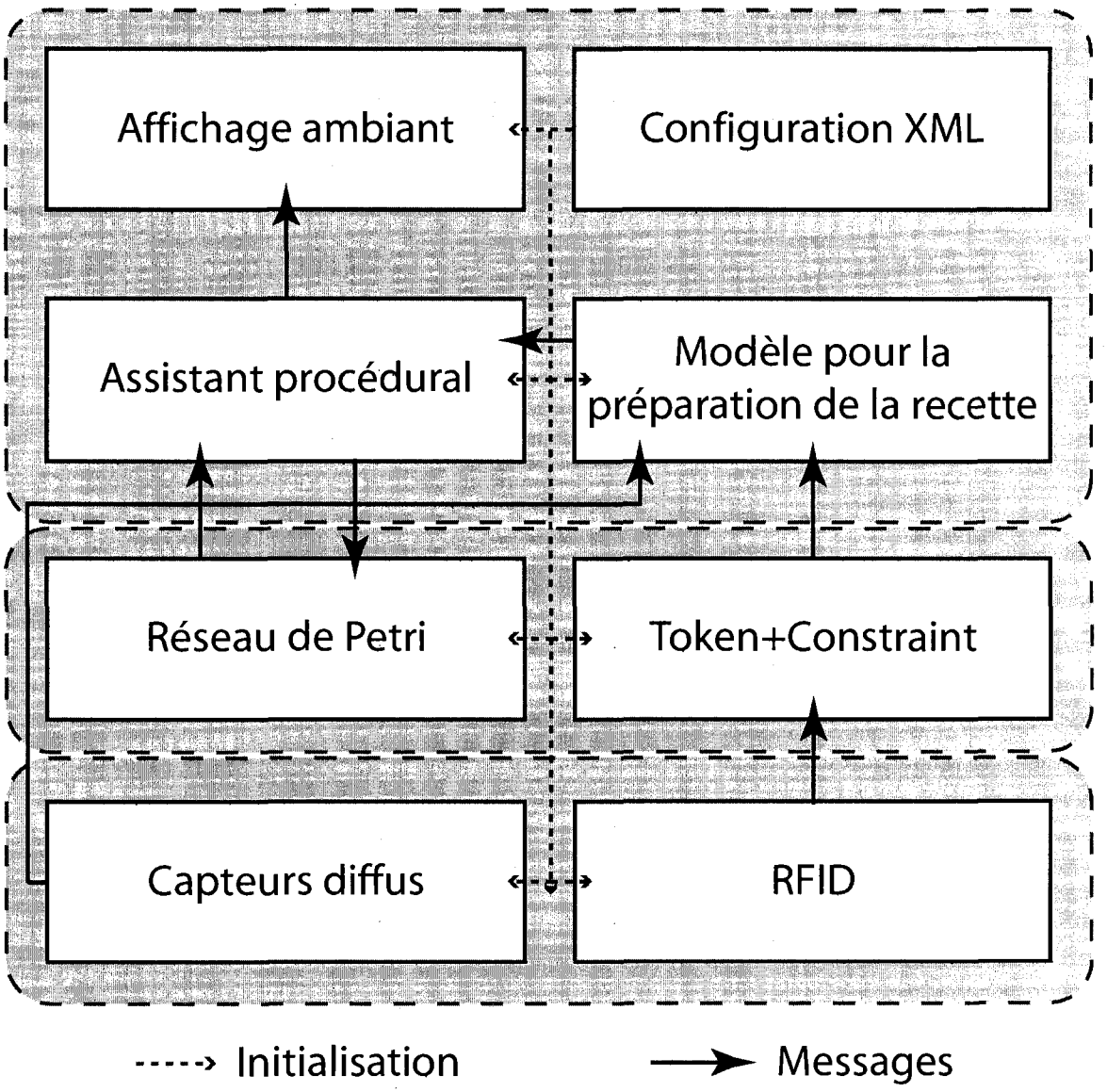

FIGURE 4.1 – Architecture logicielle pour le prototype réel

4.1 Token+Constraint (TAC)

Existe-t-il un modèle pour les interfaces tangibles sur lequel toutes nos applications puissent se baser ? Le *Token and Constraints* s'avère être le seul modèle uniforme existant actuellement sur les interfaces tangibles. Enfin existe-t-il une boîte à outils pour ce modèle ? Oui il en existe, mais certains outils ne sont pas ouverts, d'autres sont trop complexes.

C'est pourquoi j'ai créé ma propre boîte à outils fondée sur le modèle *TAC* en ayant comme objectif de coller au plus près à la définition de ce modèle. Le modèle *TAC* a été défini dans plusieurs articles [8] [62] [68]. Cette section explique la transposition dans un langage orienté objet, en l'occurrence *Java* du modèle *TAC*.

4.1.1 Le modèle TAC

TAC est un type d'interaction tangible où un objet est positionné sur une contrainte et les manipulations de celui-ci entraînent des changements dans une variable qui lui est associée. Le *TAC* formalise ces interactions. Le modèle comporte des définitions et des règles. Ce modèle a été capable de modéliser plusieurs systèmes [8] [62] comme le système *Designer's Outpost*.

Définitions

Le modèle *TAC* définit plusieurs types d'objets :

1. *Pyfo* : un objet physique ;
2. *Token* (Jeton) : un *pyfo* qui peut se positionner sur une contrainte ;
3. *Constraint* (Contrainte) : un *pyfo* qui limite les degrés de liberté des jetons ;
4. *Token+Constraint* (TAC) : l'ensemble des interactions virtuelles possibles de l'objet vers la variable ;
5. *Variable* : une information digitale associée au *TAC*.

Le *TAC* est un ensemble d'interactions. Les contraintes limitent les manipulations physiques possibles du jeton, appelées degrés de liberté. Les manipulations physiques sur l'objet entraîneront des modifications sur la variable virtuelle associée au *TAC*.

Règles

Il y a cinq règles définies par le modèle *TAC* :

1. Un *pyfo* doit être associé avec une variable et une contrainte afin d'être considéré comme jeton,
2. Chaque *pyfo* peut être défini comme contrainte ou comme jeton ou les deux,
3. Un jeton garde la liste de ses contraintes, mais la contrainte ne garde pas la liste de ses jetons,

4. Une variable est associée avec chaque *TAC*,
5. Chaque *TAC* est contrôlé par une action physique discrète ou continue.

Le modèle fonctionne de manière discrète. La couche des capteurs rend discrets les événements continus.

La différence entre continu et discret réside dans le fait que dans le continu, la sortie suit fidèlement l'entrée et que dans le discret, une impulsion de l'entrée change l'état de la sortie (fig. 4.2). Cela dépend de la conception du matériel.

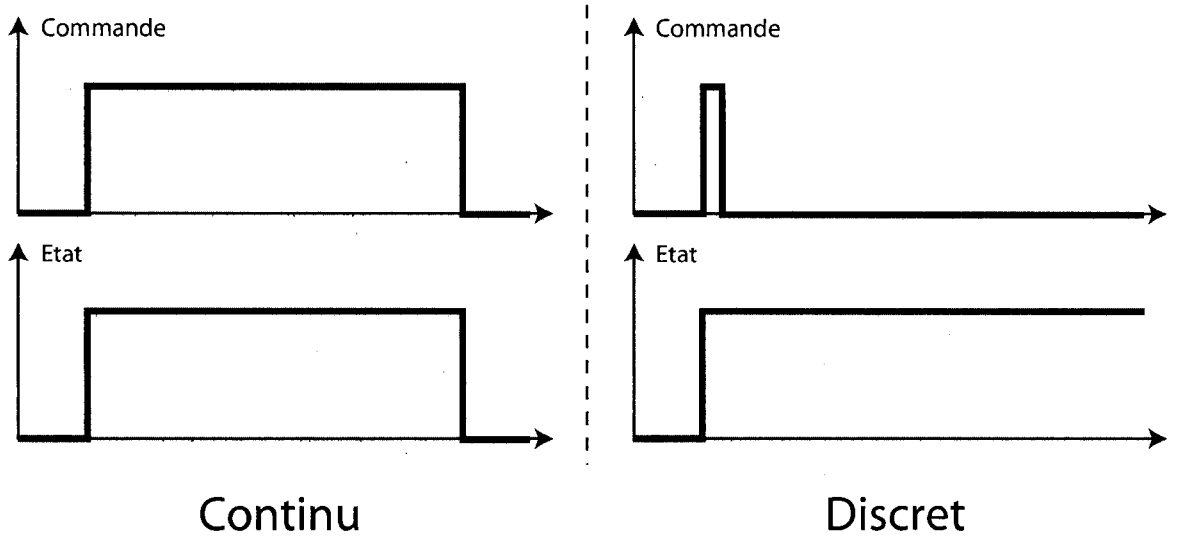

FIGURE 4.2 – Continu et discret

4.1.2 L'implantation en objets du modèle TAC

L'implantation du modèle *TAC* (fig. 4.3) respecte à la lettre les définitions données en 4.1.1.

Les classes correspondent aux concepts de contrainte (fig. 4.5), *pyfo* (fig. 4.6), jeton (fig. 4.7), *TAC* (fig. 4.9) et variable (défini en dehors du *framework TAC*). La variable ne fait pas partie du *framework TAC* car elle est spécifique à l'application finale. L'action (fig. 4.8) connaît la variable ainsi que ses méthodes.

Une classe capteur, qui implémente l'interface *Sensor* (fig. 4.10), envoie les événements matériels à la contrainte et aux actions possibles. L'interface *Sensor* force la classe capteur à accepter d'enregistrer des contraintes. Le capteur enverra des événements aux contraintes enregistrées. Le capteur modélise le fait que l'action est discrète ou continue.

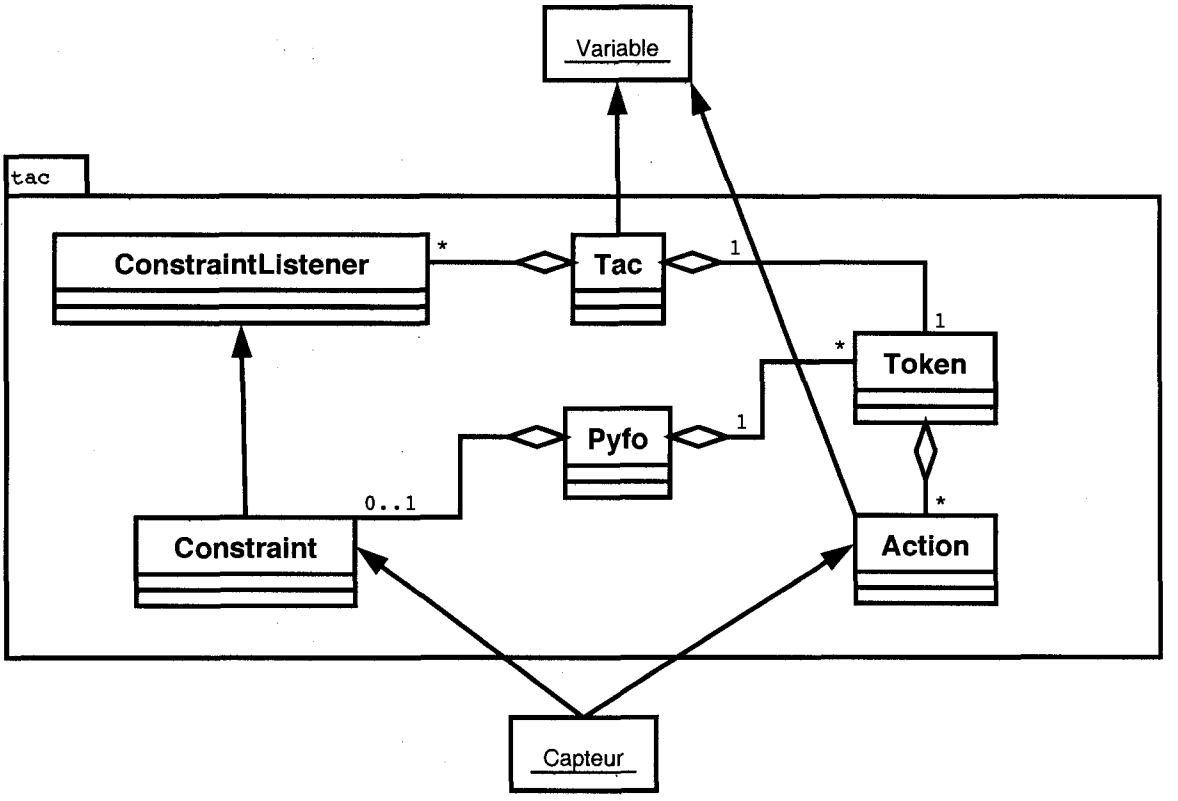

FIGURE 4.3 – Diagramme de classes UML de la couche Token+Constraint

Une classe *Action* est ajoutée par rapport à la définition originale. En effet, d'après la *règle 1*, un objet est considéré comme jeton lorsqu'il est associé à une variable et une contrainte, comme les degrés de liberté de l'objet sont connus, il est possible de déterminer l'ensemble des actions possibles sur l'objet et la variable. Cette classe action représente donc une action. Cette action est liée à l'action physique et définit quel changement il faut appliquer à la variable en appelant une méthode de celle-ci. Il peut y avoir plusieurs actions associées au jeton et cela représente donc l'ensemble des manipulations possibles sur l'objet devenu jeton. Toutes les actions physiques ne sont pas représentées, mais seulement celles utilisées pour la manipulation de la variable associée au *TAC*.

Pourquoi ai-je associé les actions au jeton et pas au *TAC* ? *Token* prend son sens avec l'ensemble des actions physiques et virtuelles. En effet, l'utilisateur manipule le jeton pour manipuler la variable. Le nom de l'action sera toujours lié avec l'action physique et l'action virtuelle faite sur le jeton et sur la variable. Le fait d'associer les actions au Token a un donc un sens, car cela exprime les métaphores possibles.

Dans l'exemple du projet Answering Machine (section 1.3.5), un message est un jeton de type "message". Les contraintes sont les suivantes [62] :

1. la contrainte de la lecture, qui accepte un seul jeton de type message,
2. la file d'attente de messages, qui accepte plusieurs jetons de type message,
3. chaque jeton "message" exerce dans la file d'attente une ou deux contraintes à un ou deux jetons "message", en amont ou en aval.
4. le réceptacle de messages vides.

Le réceptacle de messages vides contient un ensemble de jetons qui ne sont pas utilisés. Le système est capable d'associer un message à un de ces jetons et placer ce jeton dans la file d'attente. De même, l'utilisateur recycle un message en le positionnant dans ce réceptacle et le système va dissocier la variable associée au jeton "message".

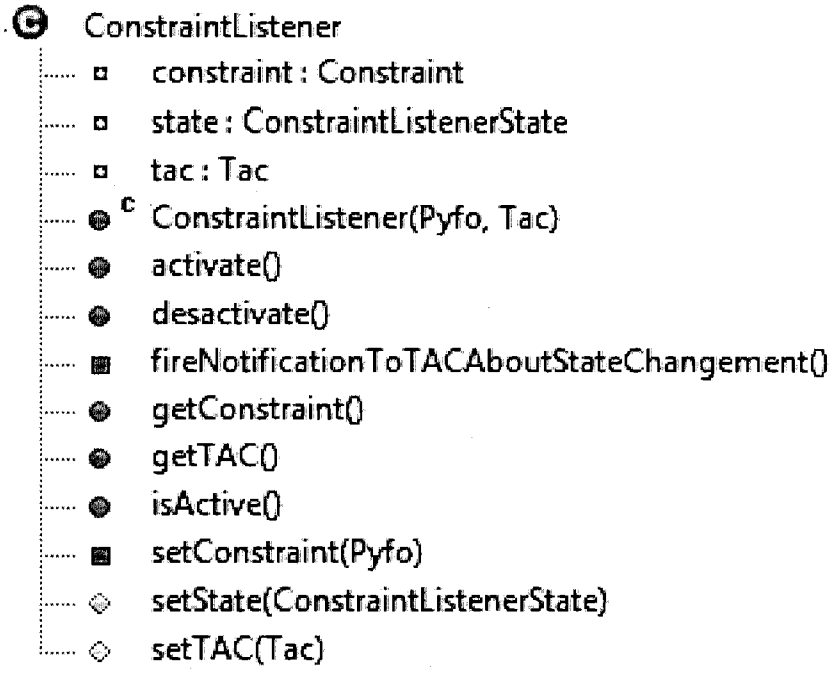

FIGURE 4.4 – Description de la classe ConstraintListener

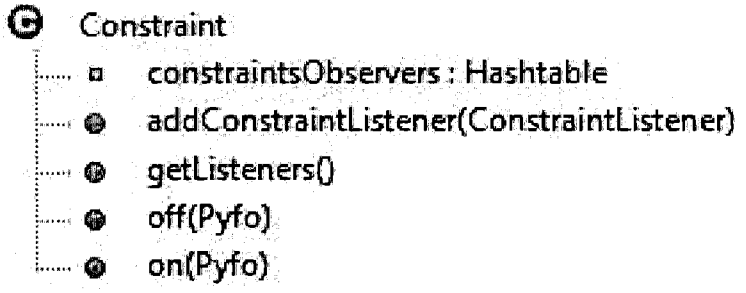

FIGURE 4.5 – Description de la classe Constraint

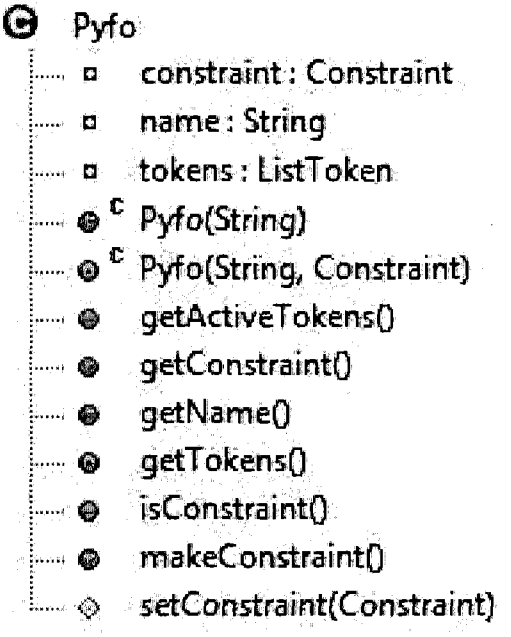

FIGURE 4.6 – Description de la classe Pyfo

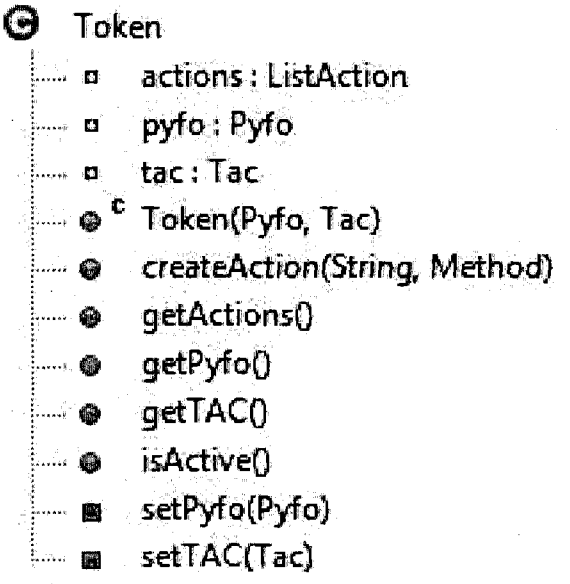

FIGURE 4.7 – Description de la classe Token

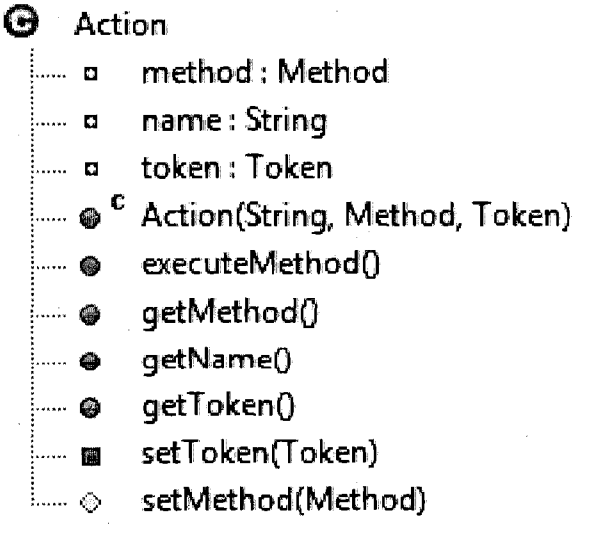

FIGURE 4.8 – Description de la classe Action

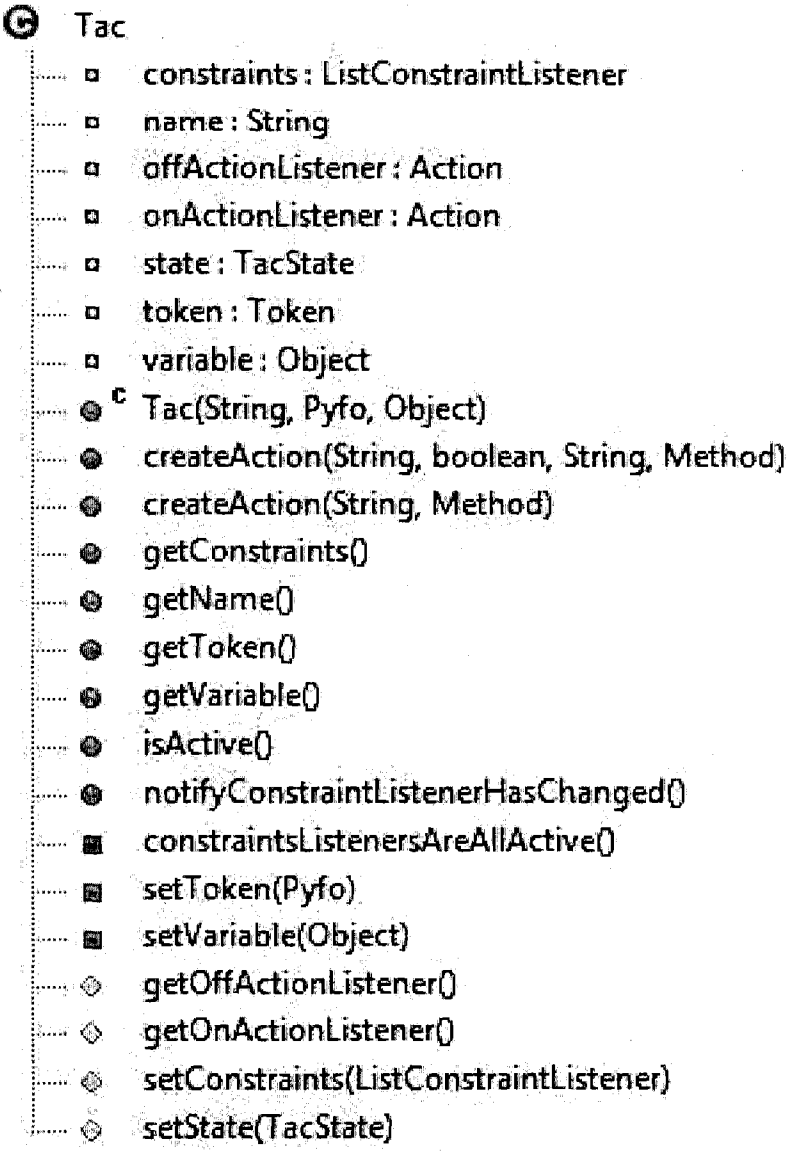

FIGURE 4.9 – Description de la classe Tac

FIGURE 4.10 – Description de l'interface Sensor

Représentation du modèle à l'aide d'un fichier XML

L'application doit être personnalisable. Il faut donc un fichier *XML* de configuration pour définir les *TAC*.

Le fichier *XML* représente très fidèlement les tableaux de l'article du TAC [8]. La modélisation des interactions reprend cette même méthode (tab. 4.1). Enfin, il a fallu ajouter la liste des instances des objets, car le tableau ne représente que des types d'objets et de contraintes.

Voir annexe A pour le résultat de la structure du fichier XML.

Le signal de retour a été supprimé du tableau, car c'est une description textuelle, c'est pourquoi il n'est pas utilisé par la couche *TAC*. Cette information est par contre utile pour la documentation de l'implantation des méthodes exécutées, comme *add*, *remove*, *up* et *down*. Un schéma *XML* (fig. 4.11) définit la grammaire du fichier de configuration.

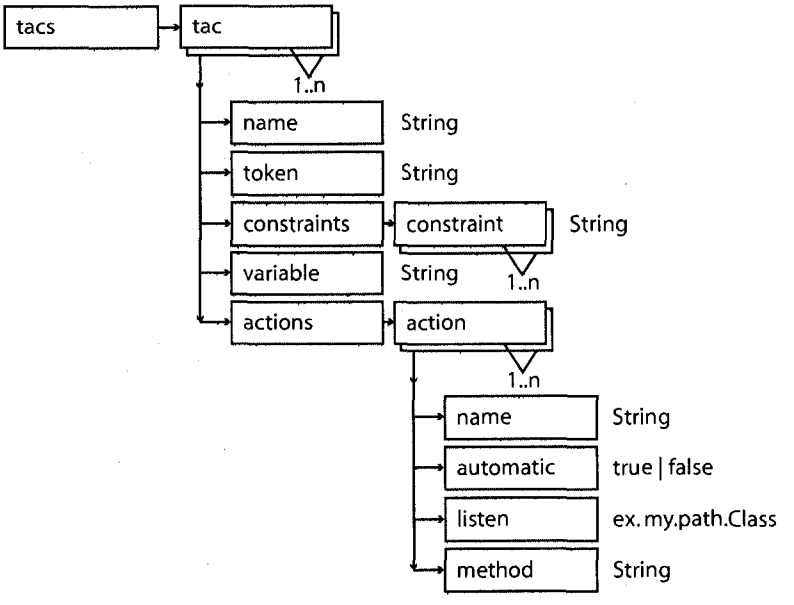

FIGURE 4.11 – Représentation TAC en XML

Le schéma définit les éléments suivant :

- Le nom du *TAC* est pour représenter la numérotation faite dans les articles, mais il est possible de mettre un nom plus compréhensible.
- *Token* est le type de *pyfo* accepté par le *TAC*.
- *Constraints* est la liste des contraintes où le pyfo doit être placé afin d'être considéré comme *token*.

TABLEAU 4.1 – Modèle TAC pour l'application simulée

TAC	Constraints	Token	Variable	Actions	Feedback
1	Help	Task	Task	Add	Affiche aide de la tâche
				Remove	Masque aide de la tâche
				Up	Défilement haut
				Down	Défilement bas
2	Get information	Task	Task	Add	Affiche infos de la tâche
				Remove	Masque infos de la tâche
				Up	Défilement haut
				Down	Défilement bas
3	Drop	Task	Task	Add	Supprime la tâche
4	Get information	Container	Content	Add	Affiche infos du contenu
				Remove	Masque infos du contenu
				Up	Défilement haut
				Down	Défilement bas
5	Pour, Destination	Container	Content	Add	Contenant va recevoir
6	Pour, Source	Container	Content	Add	Contenu va être verser
				Remove	Action Contenu versé
				Up	Augmente quantité
				Down	Diminue quantité
7	Help	Container	Content	Add	Affiche aide du contenu
				Remove	Masque aide du contenu
				Up	Défilement haut
				Down	Défilement bas
8	Drop	Container	Content	Add	Supprimer le contenu
9	Achat	Container	Content	Add	Achète un produit
10	Mix	Container	Content	Add	Mélange le contenu
11	Heat	Container	Content	Add	Chauffe le contenu
				Remove	Saisi la température
				Up	Augmente température
				Down	Diminue la température

- Variable est le chemin de la classe de la variable associée, par exemple *java.lang.String*.
- Actions est la liste des actions possible lorsque le *TAC* est actif.
- Il y a deux types d'actions, la manipulation et l'association.
- L'association (*automatic* égale à true et *name* est *add* ou *remove*) exécute automatiquement la méthode sans écouter d'événements d'une classe (*listen* est vide).
- La manipulation (*automatic* égale à *false* et *name* différent de *add* ou *remove*) exécute la méthode lorsqu'il reçoit un événement de la classe défini dans listen, par exemple *my.path.Sensor*.

Le *TAC* ne suffit pas à lui seul pour être fonctionnel, car il définit seulement les types des objets, mais pas leurs instances. J'ajoute donc la définition des instances des types *Contraintes* et *Token* (fig. 4.12).

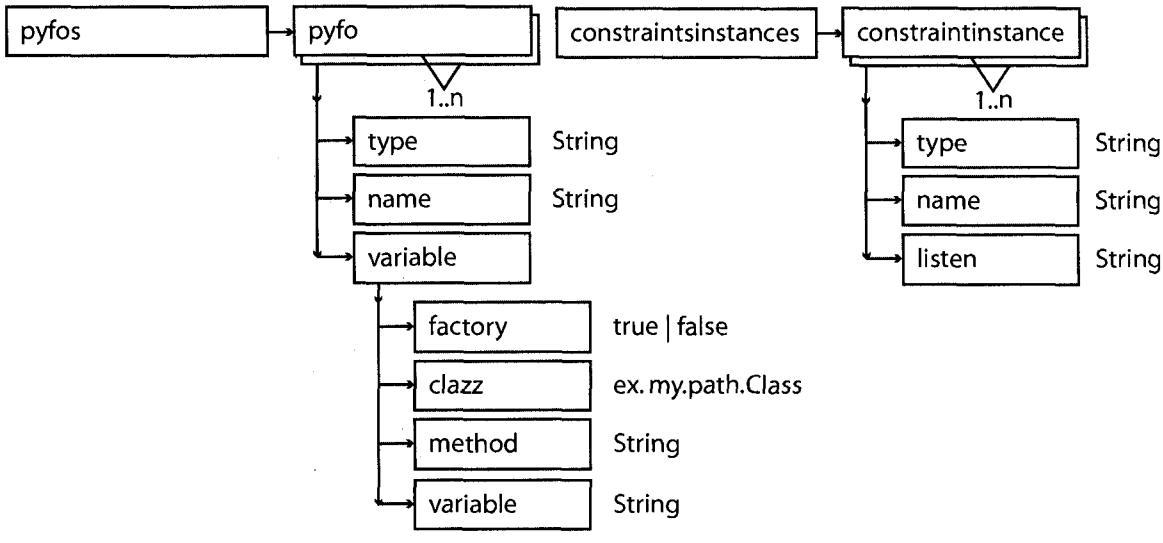

FIGURE 4.12 – Représentation des instances de TAC en XML

- *Pyfos* contient toutes les instances des pyfos qui peuvent devenir tokens.
- Le type correspond au type défini dans les *TAC*.
- Le nom permet d'identifier le *pyfo*.
- Enfin la définition de la variable associée. Le champ *clazz* défini la classe de la variable associée. Deux cas sont possible pour définir comment le *TAC* est créé ou retrouve la variable. Soit, il appelle le constructeur de la classe, auquel cas factory est *false* et *method* est ignoré. Soit, il appelle une "méthode usine" (*factory method*) qui retourne la variable, auquel cas factory est *true* et *method* est le nom

de la méthode qui sera exécutée (la méthode est sans argument).
- *Constraintsinstances* contient toutes les instances des contraintes.
- Le type correspond au type défini dans les *TAC*.
- Le nom permet d'identifier la contrainte.
- Enfin la classe qu'il écoute pour avoir les événements d'association de *Pyfo*. La contrainte possède deux méthodes que la classe qu'il écoute doit appeler : les méthodes *on* et *off* prennent toutes les deux en argument le *pyfo* qui est associé au *TAC*.

Comme un *pyfo* peut être à la fois *token* et *contrainte*, la création de l'instance *TAC* doit en tenir compte.

Un exemple de fichier de configuration est fourni en annexe (B.3)

4.1.3 Résumé du fonctionnement

1. Une *contrainte* est un élément physique (*Pyfo*) qui exerce une contrainte sur un autre élément physique,
2. Un *TAC* est la modélisation d'un type d'interaction et la relation *jeton/contrainte*,
3. Lorsqu'un objet satisfait toutes les *contraintes* définies dans un *TAC*, cet objet devient un *jeton*, permettant certaines actions sur la variable associée au *TAC*. Le système retourne alors une réponse.

TAC est une couche qui se place au-dessus des capteurs et qui modélise le comportement de l'application. C'est donc une structure de type contrôleur. L'initialisation de la couche par *XML* rend la structure hautement personnalisable. Enfin c'est un modèle qui semble être bon pour une utilisation pour l'assistance cognitive, il faudra le vérifier.

4.2 Réseaux de Petri

Les réseaux de Petri servent à modéliser des machines d'états pour des systèmes informatiques ou industriels. C'est une solution simple pour réaliser une machine d'état pour l'assistance.

Trois ensembles existent :

1. l'ensemble des places, qui contiennent des jetons,

2. l'ensemble des transitions, qui lorsqu'elles sont activées consomment un nombre de jetons défini des places en amont et génèrent un nombre de jetons défini aux places en aval,
3. et l'ensemble des arcs, qui définissent comment sont reliés les arcs et les places et le nombre de jetons consommés ou produits.

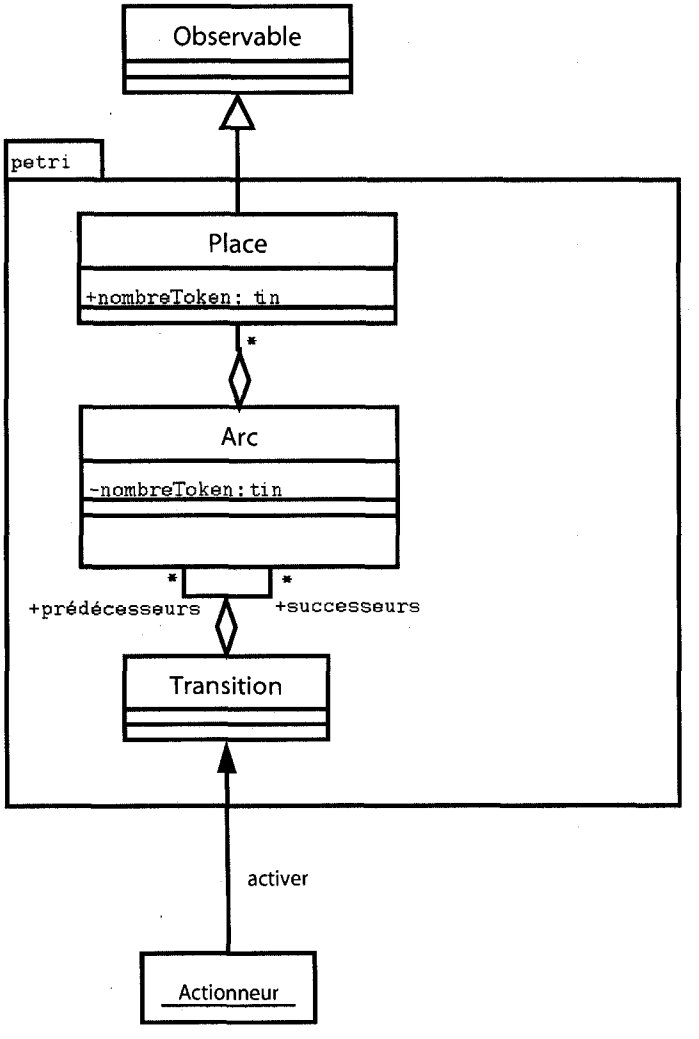

FIGURE 4.13 – Diagramme de classes UML de la couche réseau de Petri

Une implémentation simple est proposée (fig. 4.13) pour permettre n'importe quelle configuration. Dans le cas de plusieurs réseaux de Petri qui interagissent entre eux (cf. hiérarchie), des classes de contrôle doivent être créées suivant chaque cas. Il serait intéressant de créer une autre couche qui génère la possibilité de relations entre plusieurs

réseaux de Petri. Comme cela, plusieurs réseaux de Petri simple s'intègreront ensemble pour former un réseau de Petri hiérarchisé.

L'utilisateur du *framework* doit savoir activer les transitions et récupérer l'état du réseau. Pour cela la classe *Transition* possède une méthode *activate* et la classe *Place* est *Observable* afin d'être informée de tout changement dans le réseau. Pour faciliter la description et la création des réseaux de Petri, *XML* est utilisé (fig. 4.14). Un exemple de fichier XML est fourni en annexe (B.4).

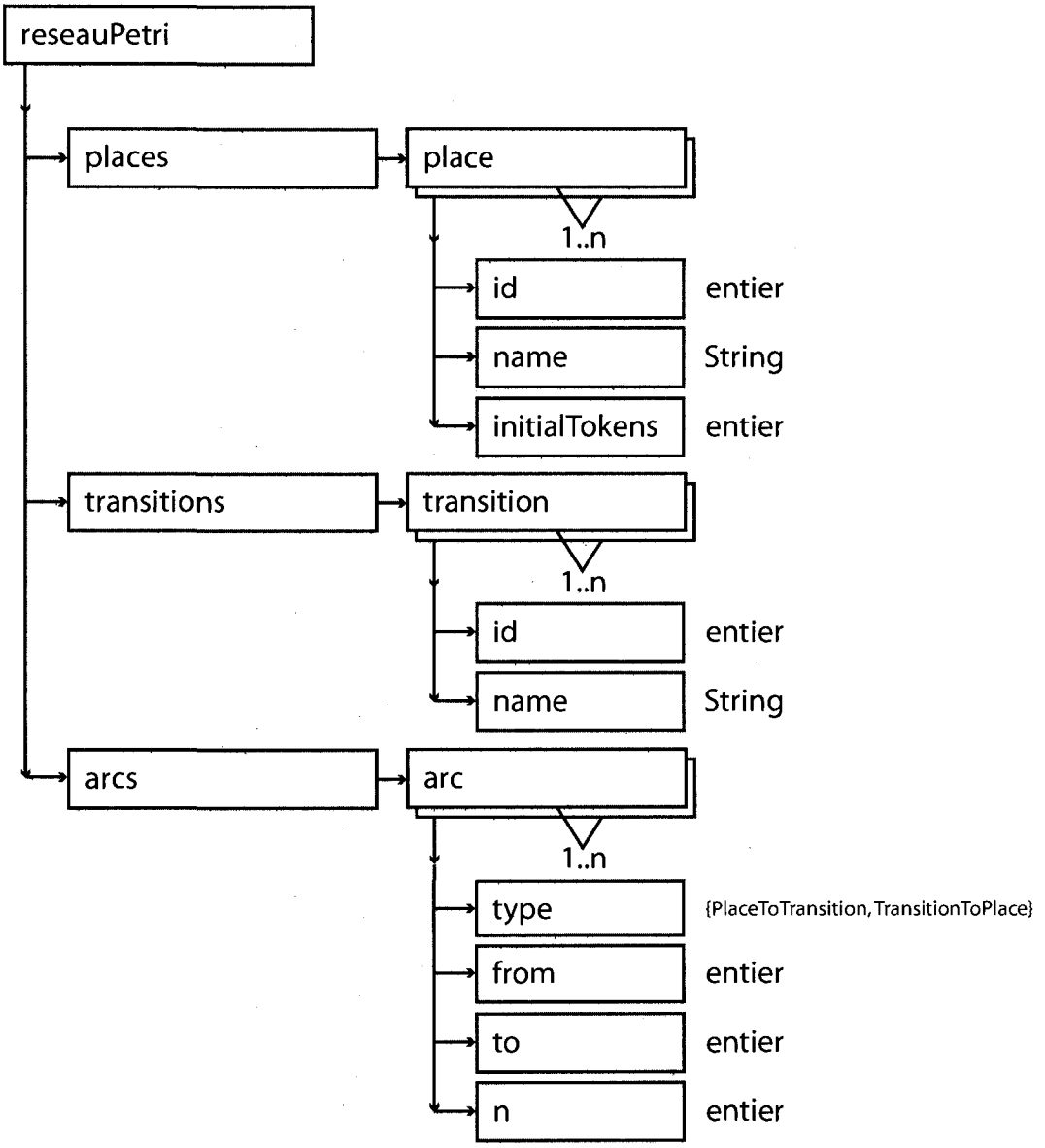

FIGURE 4.14 – Représentation d'un réseau de Petri en XML

4.3 reacTIVision

Les ajouts proposés permettent d'utiliser le *framework reacTIVision* avec le modèle *TAC*. Le but est de proposer des interactions *TAC* à l'aide d'une table interactive. Une table interactive n'a pas la notion de contrainte définie, il faut pour cela améliorer la structure de reacTIVision et de la table. Il y a deux possibilités qui se présentent alors :

1. soit il faut construire des contraintes sur la table et détecter la position de l'objet,
2. soit il faut afficher des zones représentant les contraintes.

Dans les deux cas au niveau de la détection à l'aide de *reacTIVision* et de l'affichage, la contrainte est une zone. Cette zone est une empreinte qui représente la contrainte, si l'objet est dans la zone, alors il est dans la contrainte, et inversement. C'est pourquoi je parle de zone et non de contrainte, car il ne peut pas y avoir de contrainte physique.

Pourquoi est-ce que je n'impose pas la construction de contraintes sur la table ? La vision ne permet pas de détecter la position verticale de l'objet. Un objet peut en apparence être placé dans une contrainte alors qu'en fait il est au-dessus de la contrainte. En effet, le fait d'avoir des contraintes oblige l'utilisateur à soulever l'objet, ce qui n'est pas détectable. C'est pourquoi je préfère ne pas imposer quoi que ce soit et j'utiliserai simplement une table avec des zones qui ressemblent à des contraintes. La meilleure solution serait que les contraintes soient incrustées dans la table, auquel cas il n'est plus nécessaire de soulever un objet. Par contre, la table sera moins modifiable et plus coûteuse.

J'ai donc introduit la définition de zones au *framework reacTIVision*. Un objet est soit dans une zone, soit à l'extérieur, et par conséquent dans la contrainte ou pas. De même, un objet peut être dans plusieurs zones tout comme un objet physique peut satisfaire plusieurs contraintes. Par contre, un objet sur la table ne peut pas être une contrainte. Il faudrait détecter la collision des objets et donc définir les formes des objets physiques. De plus, certains objets s'emboîtent, tel que la casserole déversant son contenu dans la passoire.

Pour utiliser la couche *reacTIVision*, un patron de conception observateur permet à toute classe, implémentant l'interface ZoneEventListener de s'enregistrer auprès d'une zone et d'être au courant de l'entrée et de la sortie d'un objet de cette zone. Ainsi *reacTIVision* est préparé pour son utilisation avec la couche *TAC*. Une interface tangible mixte est donc conçue, qui est une table interactive où les interactions sont modélisées avec *TAC*. La grande force de cette solution est de proposer une solution de prototypage rapide sans construire de contraintes physiques. De plus le simulateur de *reacTIVision*

vient aider le développement du projet, car la table n'a même pas besoin d'être réalisée pour concevoir ce prototype.

Le diagramme de classe *UML* reflète ces ajouts (fig. 4.15).

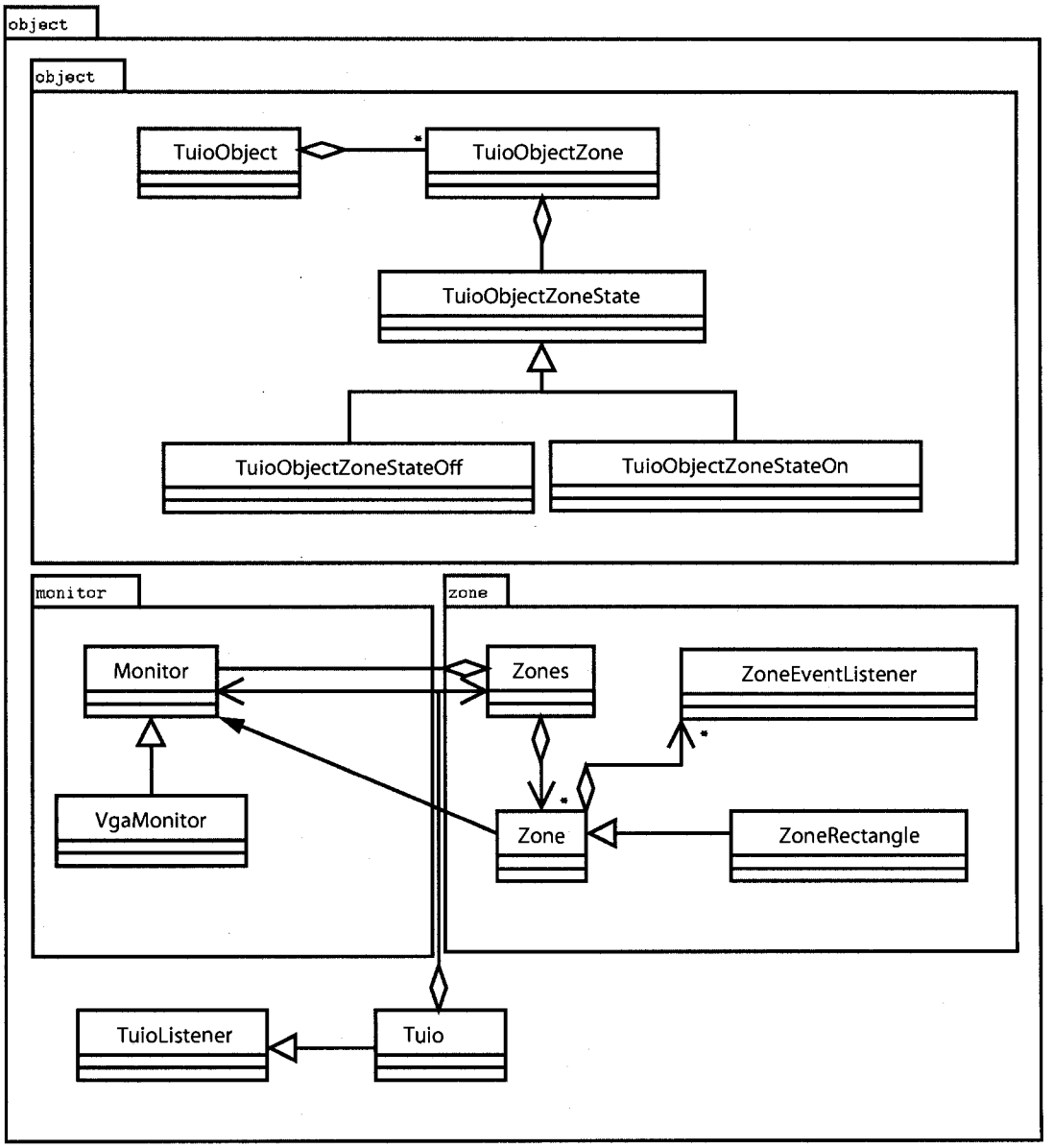

FIGURE 4.15 – Diagramme de classes UML de la couche reacTIVision

Finalement, la configuration est décrite en *XML* (fig. 4.16). Elle contient la définition de l'affichage et des zones. La zone définit la classe de l'implémentation à initialiser et les paramètres nécessaires comme le nom, la position et la taille de la zone. Un exemple

de fichier XML est fourni en annexe (B.1).

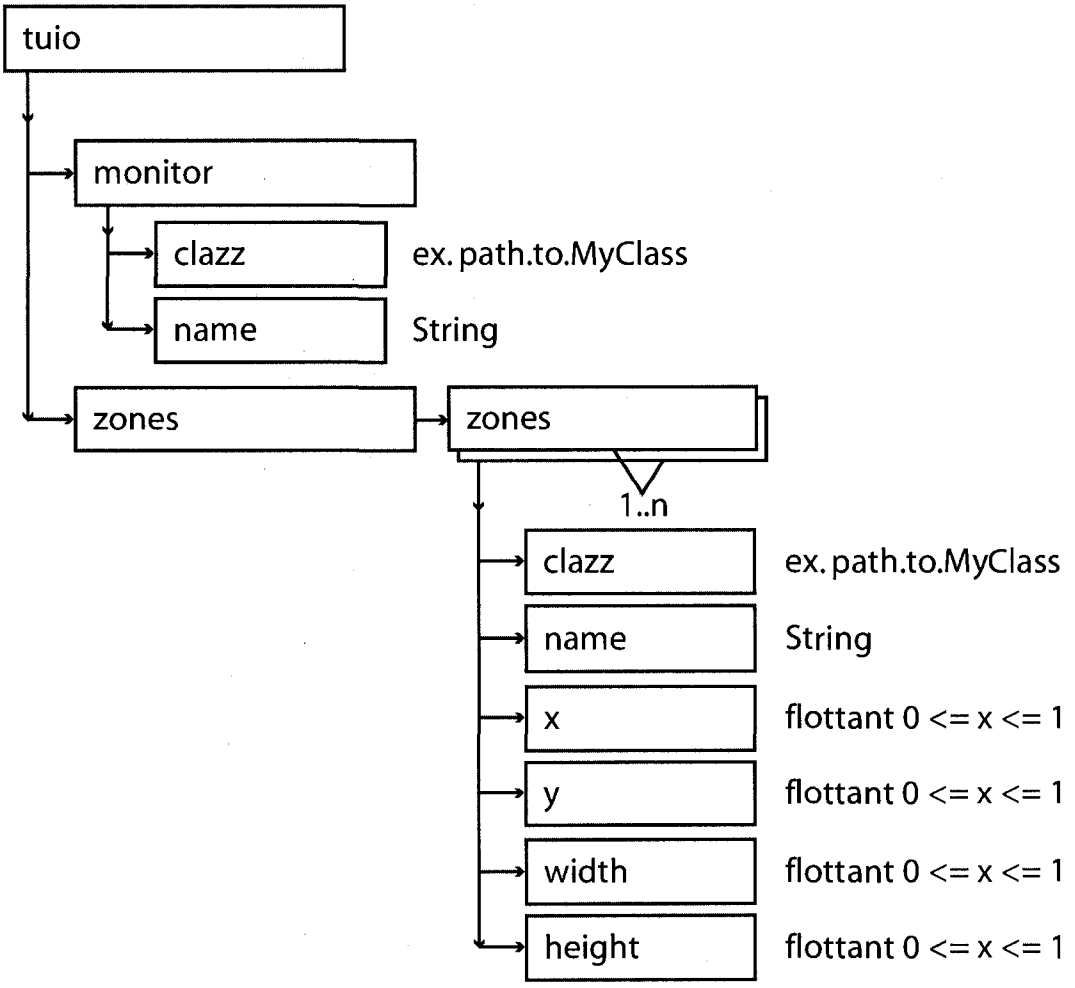

FIGURE 4.16 – Représentation des zones de la couche reacTIVision en XML

4.4 RFID

Pour notre application *RFID* est utilisés sous plusieurs formes, par exemple le *RFID* de *Phidget* et le *RFID* de *Symbol*. Ces différents systèmes ont été uniformisés pour simplifier leur utilisation (fig. 4.17).

RfidHelper est une classe qui connaît toutes les variables, ce qui définit un environnement. Les variables sont les lecteurs *RFID* de type *RfidTagReader* (indexés par nom du lecteur *RFID*) et les tags *RFID RfidTag* (indexés par identifiant du tag *RFID*).

Le *RfidTagReader* est une classe abstraite que chaque technologie doit implanter. Il est *Serializable*, pour une utilisation avec *RMI*, *Observable* et *Runnable*. Le *RfidTagReader* ne possède pas de *RfidTag*, toutefois une méthode permet de construire la liste des tags détectés par le lecteur en passant par le *RfidHelper*. Cette méthode est rarement utilisée, ce qui permet d'utiliser exceptionnellement une boucle de recherche dans une liste.

Le *RfidTag* connaît les lecteurs sur lesquels il est détecté. Comme il est *Observable*, sa faculté la plus importante, tous les observateurs sont au courant de la sortie et de l'entrée de ce tag dans tous les lecteurs connus. Il est aussi *Serializable* pour une utilisation avec *RMI*.

L'initialisation de cette couche par un fichier *XML* (fig. 4.18) permet de définir les tags *RFID* et les lecteurs de tags *RFID*. Grâce au champ *clazz* du lecteur *RFID*, il est possible de changer l'implantation utilisée. Les variables correspondent à celles de la construction. *Order* doit être *0*, puis *1*, etc. Il faut donc que l'implantation ait un constructeur acceptant pour arguments un *RfidHelper* pour le premier argument, puis un nombre allant de *0* à *n* de *String*.

Enfin, j'ai défini deux lecteurs virtuels qui permettent de simuler des lecteurs *RFID*. Le premier lecteur *SimulatedRfidTagReader* prend à un intervalle régulier un tag qui entrera et sortira de ce lecteur aléatoire.

Le deuxième lecteur *ScenarioRfidTagReader* est initialisé avec un scénario, dont la définition *XML* est défini à la figure 4.19. Le temps est relatif, ce qui a des avantages et des inconvénients. C'est plus facile à programmer et à répercuter un changement, mais c'est plus difficile à lire, car il faut additionner tous les temps d'attente pour connaître à quel moment l'action se déroulera. Le lecteur prend un argument qui est l'adresse absolue du fichier de configuration en tant que ressource. Un exemple de fichier XML est fourni en annexe C.1.

4.5 Capteurs diffus

Le laboratoire dispose de capteurs diffus qui permettent de détecter l'ouverture de tiroirs et de placards, et des capteurs de débit d'eau pour les robinets de la cuisine. Tous ces capteurs sont connectés à deux ensembles *Cresnet Pro2* et *PLC*, qui traitent des entrées et des sorties et qui sont connectés en réseau. Un serveur d'événements rassemble toutes les fonctions. Notre programme peut écouter les événements en se connectant et en s'enregistrant sur ce serveur. Pour cela une interface *EventFromServerListener* doit

être implantée et s'enregistrer à une classe *Client*. Tout cela est dans le module *domus-event-client*.

Enfin, un fichier de configuration *client.properties* est nécessaire et doit être placé comme ressource dans le dossier *configuration.io.events.client*. C'est un fichier de propriétés *Java* qui possède deux champs, *Client.port* et *Client.hostname*, qui sont respectivement le port et l'adresse du serveur.

4.5.1 Placards

Les portes de placards ou les tiroirs sont soit ouverts, soit fermés. Chaque placard est identifié par l'entrée et par la machine qui capte les informations d'ouverture et de fermeture. Enfin pour identifier l'événement, qui est une chaîne de caractères, un code d'ouverture et de fermeture est nécessaire.

L'événement est pris en compte s'il contient le code de la porte. Enfin, l'événement sera traité suivant qu'il contient le code de fermeture ou d'ouverture. Cela est en attendant qu'une description de l'environnement soit définie et c'est donc sujet à changement.

Le modèle actuellement utilisé est défini à la figure 4.20.

Pour configurer les portes de placards, un fichier XML a été défini à la figure 4.21. Un exemple de fichier XML est fourni dans l'annexe C.2.

4.5.2 Capteurs de débit d'eau

Les robinets peuvent s'ouvrir et se fermer. Le débit et la quantité d'eau sont ajoutés et calculés par chaque débitmètre. Les capteurs de débit d'eau sont identifiés par un simple nom.

Le modèle de classes UML est représenté par la figure 4.22.

Un affichage *Swing* a été repris d'une application conçue par un étudiant. Tout a été refactorisé, car une seule classe existait à l'origine pour quatre capteurs.

Pour définir les capteurs de débit, un fichier *XML* a été défini (fig. 4.23). Un exemple de fichier XML est fourni dans l'annexe C.3.

4.6 Affichages ambiants

Les affichages servent à capter l'attention et à informer l'utilisateur. Un modèle pour la prise de l'attention a été décrit dans la section 1.1.1. Je pense qu'il faut prendre en

compte ces points dans l'assemblage des interfaces. C'est pourquoi le modèle que je propose ne supporte pas un gestionnaire des messages à transmettre avec différents niveaux d'importance. Par contre, je propose un affichage composite, c'est-à-dire un affichage qui contient plusieurs affichages. C'est grâce à cet affichage composite que je suis capable de respecter les conseils pour réaliser la séparation en différents niveaux, c'est-à-dire l'abstraction, le niveau de notification et la transition (voir section 1.1.1).

Premièrement, j'ai défini une interface commune pour tous les affichages. La classe *Output* défini trois méthodes dont la plus importante est *display(String value) throws Exception.* Je n'impose aucune règle quant à l'interprétation de la chaîne de caractères value. Suivant l'implémentation, il est possible de proposer n'importe quel type d'affichage.

Je propose plusieurs interfaces ambiantes qui peuvent être assemblées et dupliquées suivant les souhaits de l'utilisateur. Les premières interfaces de sorties proviennent de la part de Phidget. Phidget propose des modules *USB* assemblables.

Le *PhidgetTextLCD* est un affichage ayant deux lignes de 20 caractères. Il est très pratique pour afficher du texte et permet de supprimer l'écran. Toutefois, il a fallu travailler l'interface proposée afin de permettre l'affichage de texte de plus de 20 caractères. Pour cela je propose une translation du texte au fur et à mesure, et un rebouclage au début du texte lorsque la fin du texte est atteinte. Sur la deuxième ligne est affichée la position du texte affiché par rapport au texte complet.

Le *PhidgetServoMotor* est capable de positionner un axe à un angle précis. Comme il a été proposé dans l'article sur les *Phidgets*, je propose un affichage qui ressemble aux cadrans d'une voiture. J'ai défini des zones qui sont activables et une position relative dans cette zone.

Enfin, je défini la classe *Outputs* qui stocke une liste d'affichages. Un fichier *XML* décrit la liste des affichages utilisés. Ainsi lors de l'initialisation à partir de ce fichier *XML* (fig. 4.24), de nombreux affichages seront initialisés en un seul coup. Un exemple de fichier XML est fourni en annexe C.7.

4.6.1 Affichage composite

L'affichage composite est constitué de plusieurs affichages. Sa fonction est de spécifier l'envoi, ou non, d'un message aux interfaces décrites lors de sa réception. L'information qui est envoyée peut être différente pour chaque interface.

Cela permet de répondre à plusieurs problèmes en permettant :

- un niveau d'abstraction,
- un filtrage des messages,
- l'accès à plusieurs affichages à partir d'une seule instance,
- de regrouper un grand nombre d'affichages.

Le champ *xml* est l'adresse du fichier *XML* contenant la liste des affichages. Le fichier *XML* utilise la définition ci-dessus pour l'initialisation de plusieurs affichages. Le nom de l'affichage sert d'identifiant pour retrouver l'affichage correspondant.

Pour cela j'ai défini un fichier de configuration *XML* (fig. 4.25).

4.7 Assistant pour la cuisine

Cette section présente les points communs entre les deux applications : la cuisine virtuelle et la cuisine réelle. Elle présente surtout l'articulation entre plusieurs modules et le fonctionnement d'une l'assistance simple (fig. 4.26).

Premièrement, les classes *ConstraintSensor* et *ActionSensor* sont abstraites et définissent comment les capteurs sont liés au TAC. *ConstraintSensor* défini le comportement pour l'association d'un *Pyfo* à une contrainte. *ActionSensor* défini le comportement pour la manipulation d'un jeton. Cela définit juste la structure, il reste que c'est l'implémentation des capteurs qui va définir les actions à faire.

Deuxièmement, la classe *Assistance* lie l'affichage ambiant, le réseau de Petri, et à une liste chaînée de "Handlers". La classe *Handler* attrape un *Message* et teste avec le *Message* étalon associé. S'il est identique au *Message* associé, alors le *Handler* active la transition du réseau de Petri associée. Dans tous les cas il envoie le *Message* à son successeur, qui lui-même l'enverra à un successeur, jusqu'à ce que *null* soit atteint. Les variables associées au *TAC* et l'instance d'*Assistance* possèdent le point d'entrée de la liste chaînée et peuvent envoyer les événements.

Enfin, la classe *Aide* est enregistré en tant qu'*Observer* auprès de chaque *Place* du réseau de Petri. Ainsi lorsqu'un jeton du réseau de Petri entre dans une place, le nom de la place est affiché. Grâce à un affichage composite, il est possible d'adapter ce message à n'importe quel affichage associé. Cette structure fonctionne parce qu'il existe seulement un jeton dans le réseau de Petri.

Cet ensemble définit un assistant simple qui fonctionne au-dessus du modèle TAC. Il n'est pas intelligent et ressemble plus à un assistant procédural qu'à un assistant cognitif. Les variables de l'environnement associées au TAC sont définies dans les modèles cuisine

virtuelle (section 4.8) et cuisine réelle (section 4.9).

4.8 Cuisine virtuelle

Afin de valider rapidement le choix du modèle TAC, j'ai modélisé les actions primaires que tout le monde effectue dans une cuisine. Lors de la préparation d'une recette, la personne effectue les tâches suivantes : verser, cuire, passer, couper (fig. 4.27). Des actions d'assistance sont disponibles comme obtenir des informations ou une aide sur tel ou tel objet. Le but est de produire une table interactive utilisant TAC avec ce modèle simplifié de préparation de recette et de valider la structure complète du programme.

Les zones affichées servent de contraintes pour le modèle TAC (cf. 4.1). Pour verser du contenu, le contenant doit tout d'abord être placé, puis le contenu que l'on veut verser. En tournant le contenu dans le sens des aiguilles d'une montre, on augmente la quantité qui sera versée. L'action de verser est effective lorsque le contenu sort de la contrainte.

Au niveau logiciel, c'est le dernier module qui définit notre application finale. Ainsi, ce module contient tous les joints entre les modules et leurs configurations (annexe B). La couche est exécutable comme application *Java* ou *Bundle* à l'aide d'une classe qui contient toutes les phases d'initialisation des autres couches.

La couche définit les classes utilisées par *TAC* et lient d'autres couches comme *reacTIVision* pour le matériel et *kitchenaid* pour les variables. *VariableHelper* et *HardwareHelper* stockent les objets utiles pour l'initialisation et l'utilisation de l'application (fig. 4.28).

ContentFactory et *TaskFactory* sont utilisés par l'initialisation *TAC* pour la création des variables associées aux *TACs*.

La figure 4.28 présente une liste non exhaustive des classes capteurs qui envoient des informations à la couche *TAC*. Ces classes sont instanciés par l'initialisation du modèle *TAC* et doivent donc être capables de se lier à la couche *reacTIVision*. *HardwareHelper* aide justement à faire ces liens.

Deux design patterns (*Composite* et *State*) modélise l'état de la préparation de la recette de cuisine. Les objets *Content* sont associés aux *TACs* qui manipule des contenus. Le patron d'état sert à changer les méthodes suivant l'emplacement du jeton contenant sur la table interactive. Par exemple, il existe l'état *InformationDisplayed* qui va autoriser l'utilisation de l'ascenseur de l'information.

4.9 Cuisine réelle

Une fois que le prototype du cas virtuel fut fonctionnel, j'ai pensé reprendre les mêmes éléments pour réaliser cette application réelle. Mais il s'est avéré plus facile de proposer d'autres interactions pour atteindre le même but. Comme les couches sont configurables, il aurait été dommage de compliquer la structure de l'application en voulant absolument utiliser les mêmes fichiers de configuration (annexe C).

Ce module est très similaire à celui du cas virtuel. La couche matérielle contient deux modules qui remplacent *reacTIVision* : un module *RFID* et un module de capteurs diffus. Enfin, les fichiers de configurations sont différents. Ainsi, l'architecture devient telle qu'il a été défini dans la figure 4.1.

Les variables associées aux TACs sont : *Pan*, *Strain*, *Glass*, *Rice* (fig. 4.29, tab. 4.2). *VariableHelper* aide encore à retrouver les instances de ces classes et à associer le rootHandler pour le réseau de Petri.

Les contraintes utilisées sont : *Pan*, *Strain*, *Glass*, *Cupboard*, *Kettle*, *Washbasin* (fig. 4.29). On remarque que le modèle utilise la fonctionnalité qu'un *Pyfo* peut être à la fois contrainte et jeton. Pour détecter l'association, les capteurs sont différents. Encore une fois la classe *HardwareHelper* aide à retrouver les instances de ces classes créées lors de l'initialisation du modèle *TAC*.

Cupboard, *Kettle*, *Washbasin* sont les lecteurs RFID qui lisent une zone correspondant aux placards, à la cuisinière et à l'évier. *Pan*, *Strain*, *Glass* sont des tags RFID qui cherchent à savoir quel autre tag RFID est présent dans cette même zone. Si un autre tag est dans la même zone que la passoire positionnée dans une zone, alors ce tag est associé à la contrainte passoire. Avec le réseau de Petri, le programme contraint la passoire à être dans l'évier avant de recevoir la casserole.

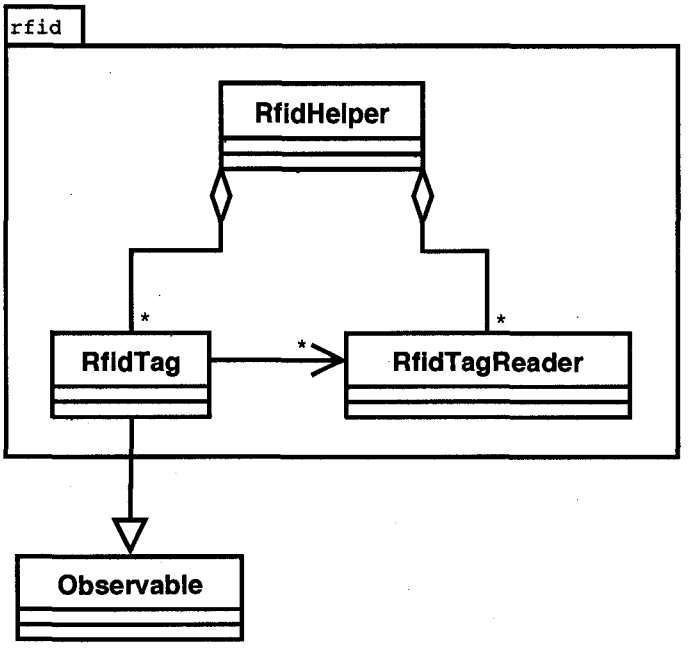

FIGURE 4.17 – Diagramme de classes UML de la couche RFID

TABLEAU 4.2 – Modèle TAC pour l'application réelle

TAC	Constraints	Token	Variable	Actions	Feedback
1	Washbasin	Pan	Pan	Add Remove	enterWashbasin goingOutWashbasin
2	Stove	Pan	Pan	Add Remove	enterStove goingOutStove
3	Cupboard	Pan	Pan	Add Remove	enterCupboard goingOutCupboard
4	Washbasin	Strain	Strain	Add Remove	enterWashbasin goingOutWashbasin
5	Strain	Pan	Pan	Add	strain
6	Cupboard	Strain	Strain	Add Remove	enterCupboard goingOutCupboard
7	Pan	Glass	Glass	Add	enterPan (verser)
8	Glass	Rice	Rice	Add	enterGlass (verser)
9	Cupboard	Rice	Rice	Add Remove	enterCupboard goingOutCupboard
10	Cupboard	Glass	Glass	Add Remove	enterCupboard goingOutCupboard
11	Washbasin	Glass	Glass	Add Remove	enterWashbasin goingOutWashbasin

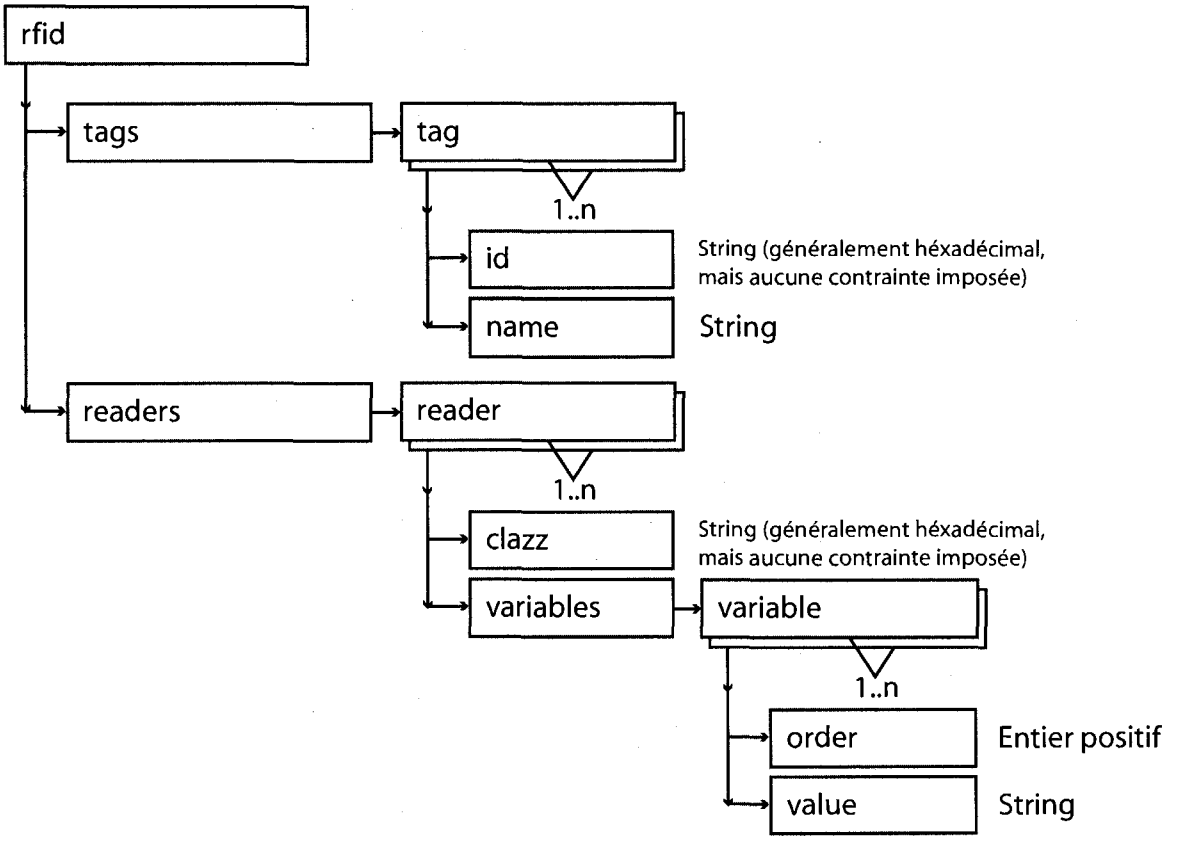

FIGURE 4.18 – Représentation des tags et lecteurs RFID en XML

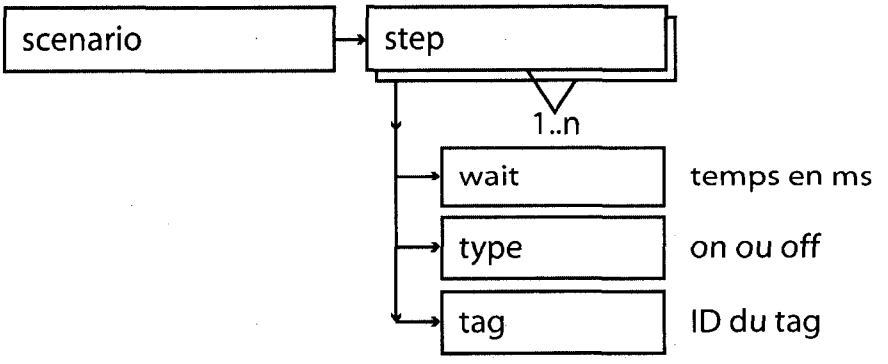

FIGURE 4.19 – Représentation du scénario en XML

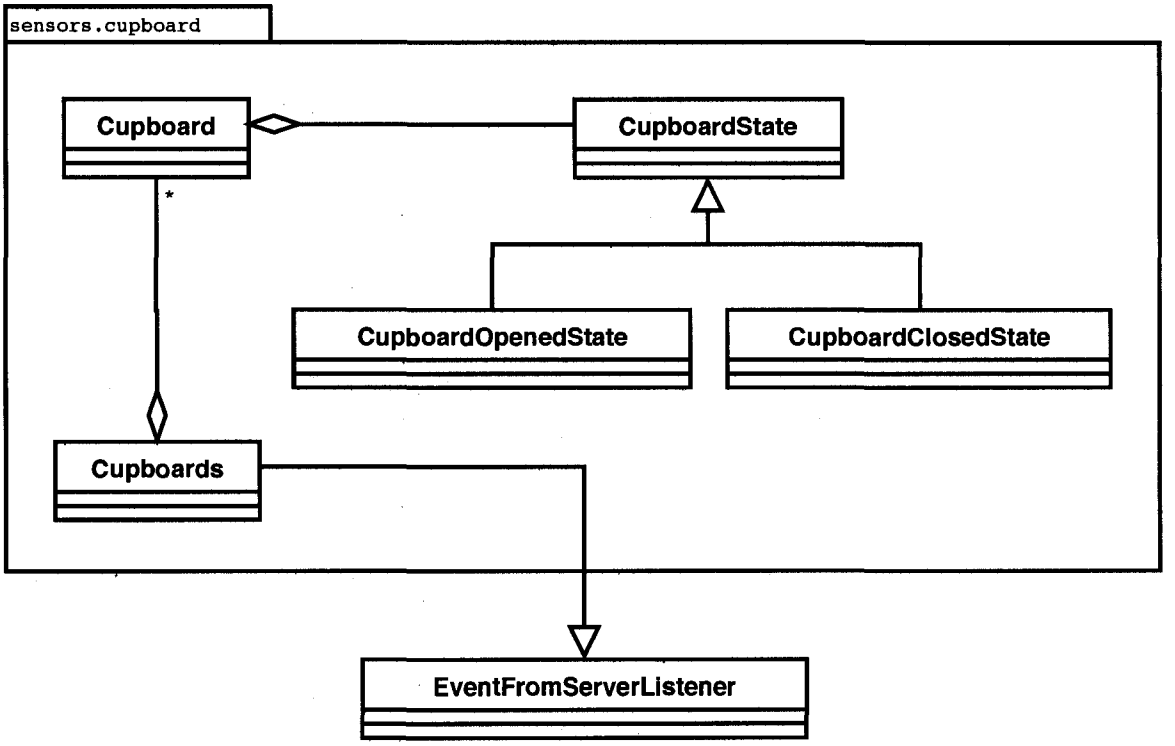

FIGURE 4.20 – Diagramme de classes UML de la couche des capteurs pour les placards

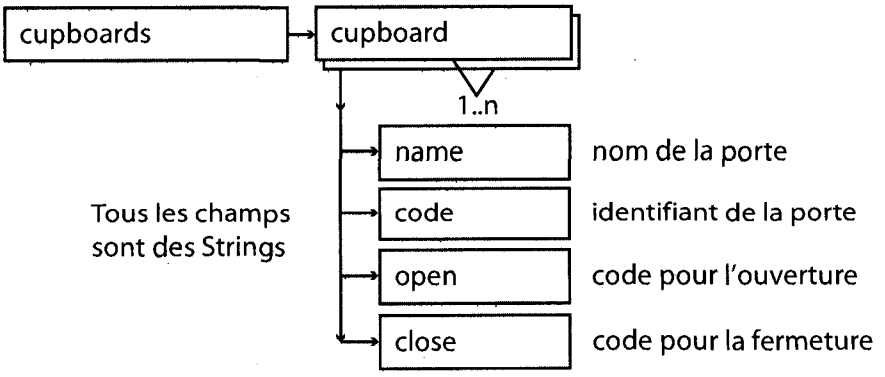

FIGURE 4.21 – Représentation des portes de placards en XML

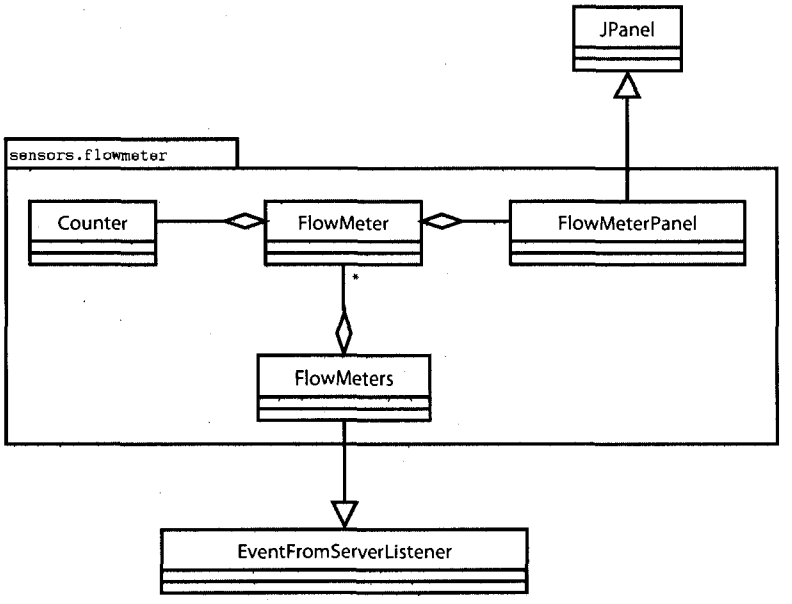

FIGURE 4.22 – Diagramme de classes UML de la couche des capteurs pour les débitmètres

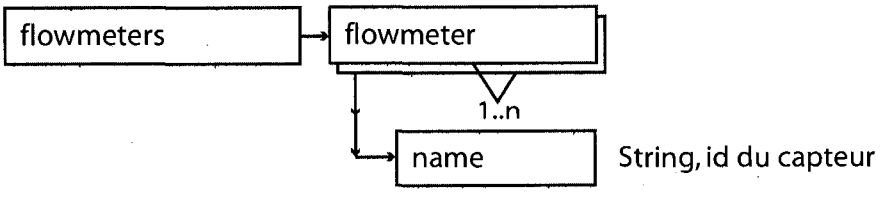

FIGURE 4.23 – Représentation des capteurs de débit d'eau en XML

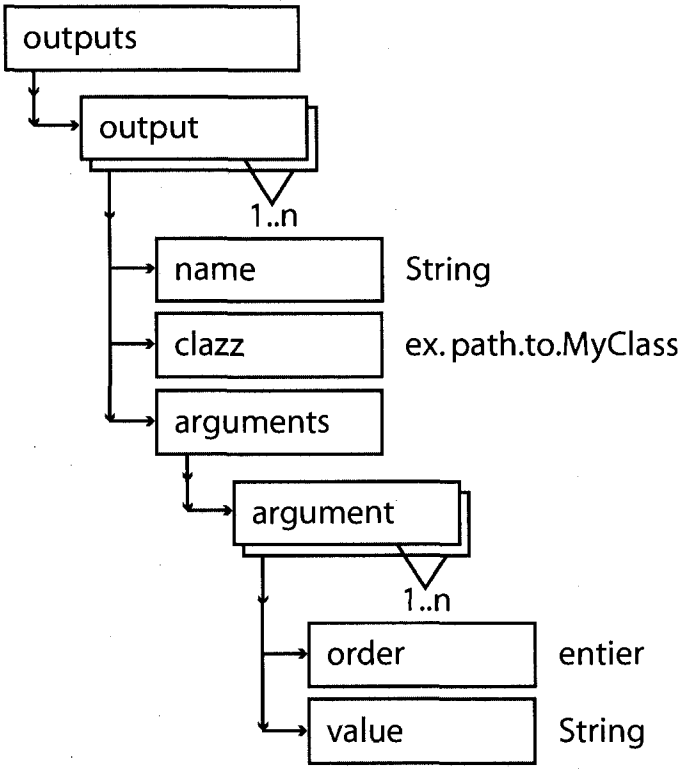

FIGURE 4.24 – Représentation des affichages en XML

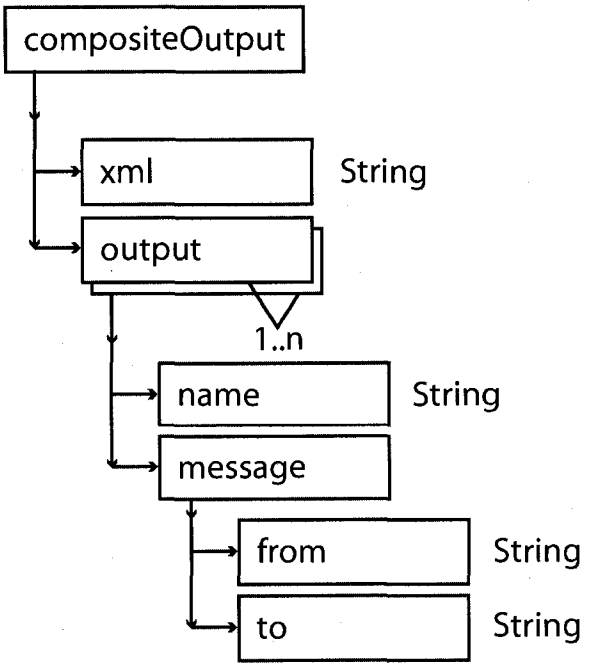

FIGURE 4.25 – Représentation de l'affichage composite en XML

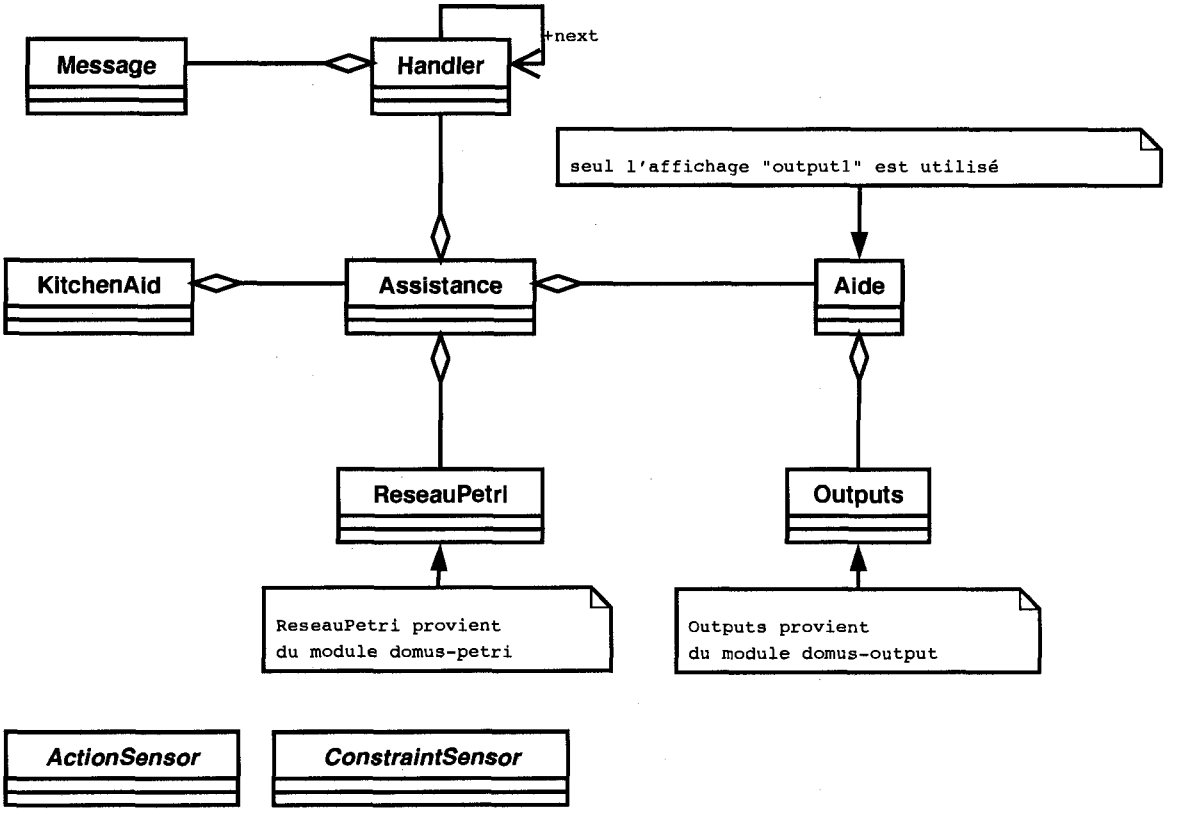

FIGURE 4.26 – Représentation de l'architecture de l'assistance commune aux deux applications

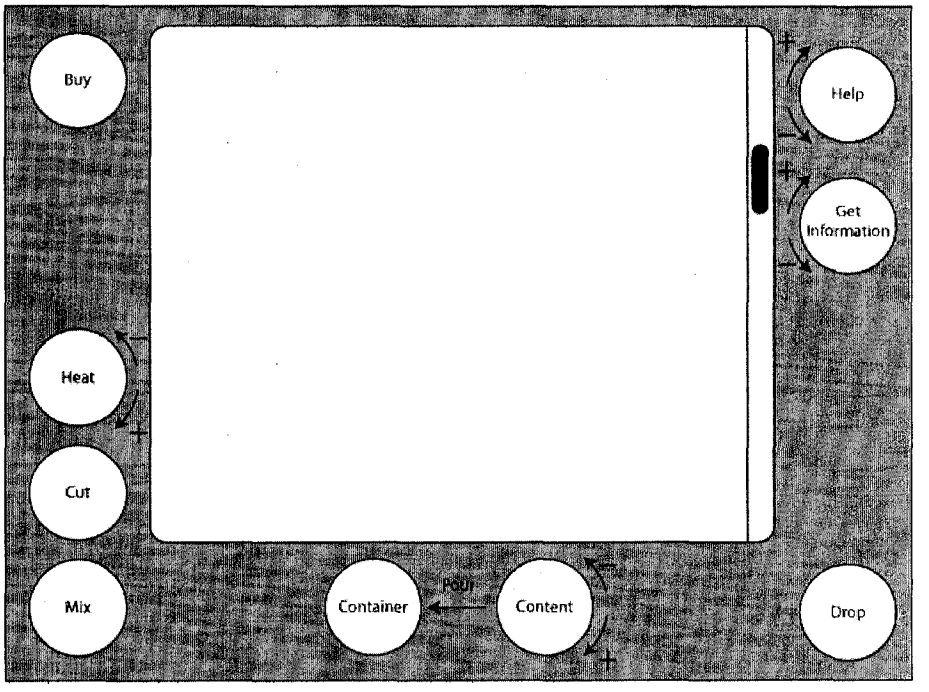

FIGURE 4.27 – Zones affichées sur la table interactive

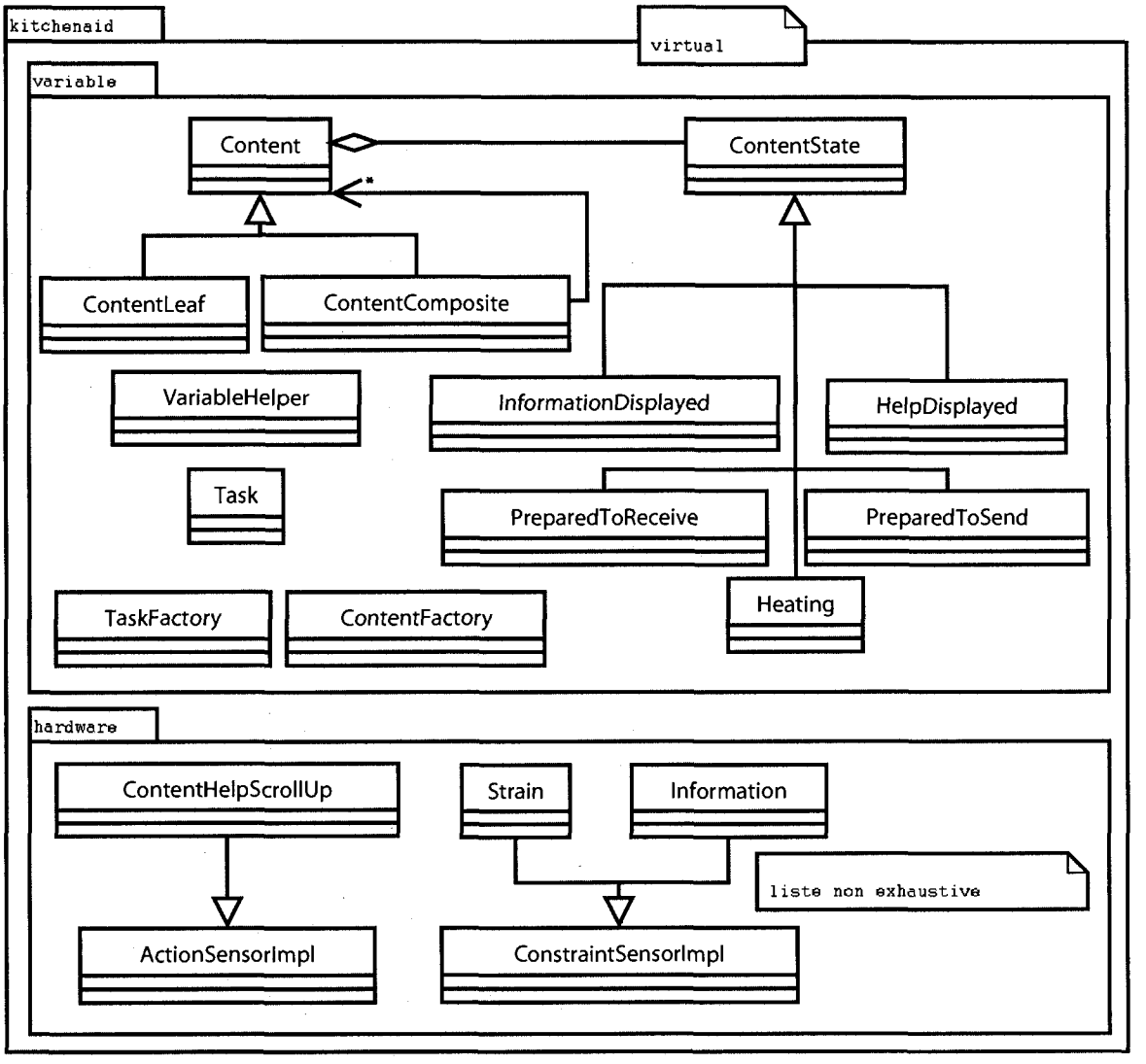

FIGURE 4.28 – Diagramme de classes UML pour les classes liant la couche TAC

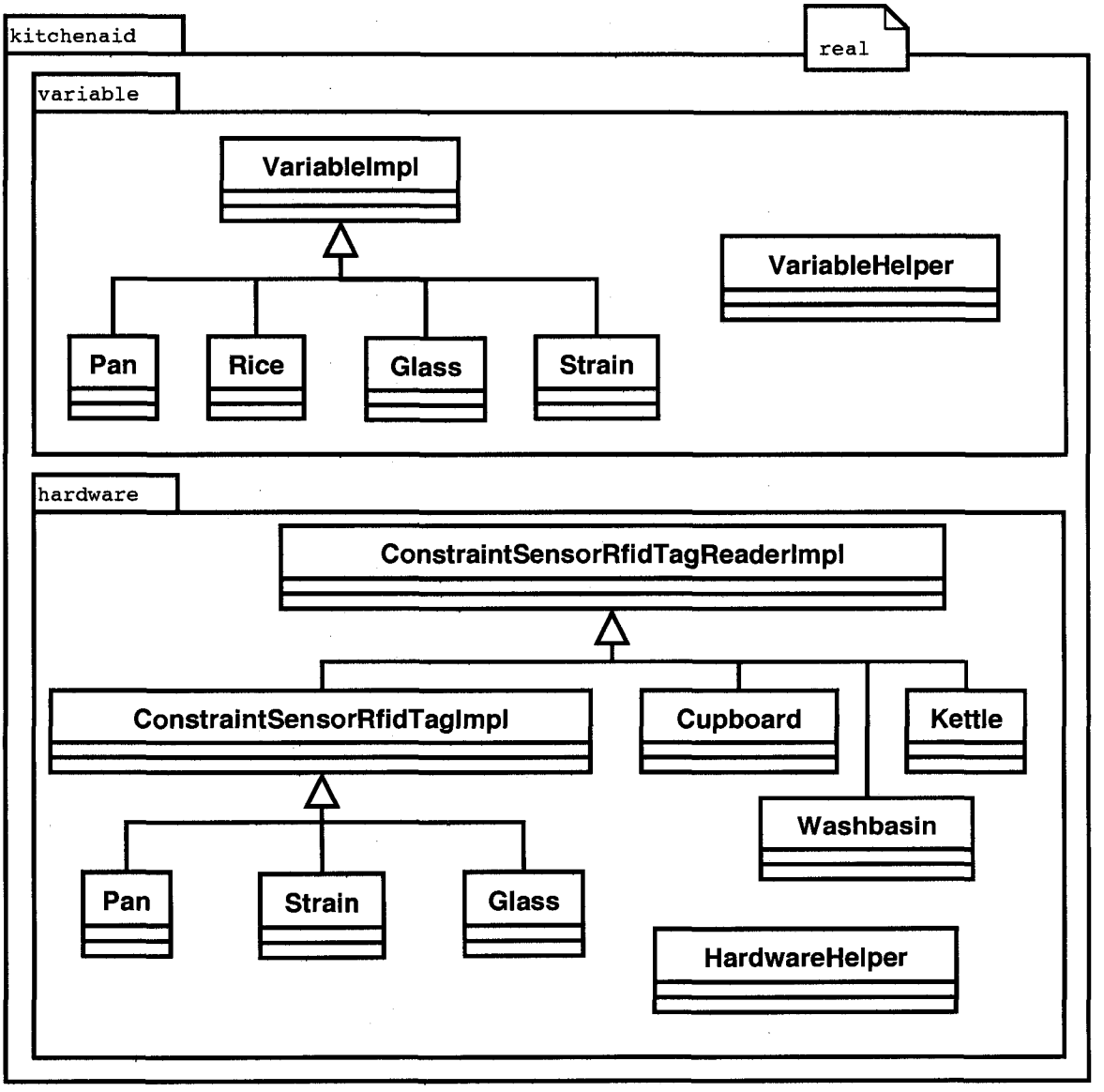

FIGURE 4.29 – Diagramme de classes UML de la couche KitchenAid réelle

Chapitre 5

Réalisations et résultats

Ce chapitre traite des réalisations des deux prototypes et des résultats obtenus. La section 5.1 fait un retour sur la méthodologie prise. Nous traitons ensuite des résultats sur l'architecture logicielle et sa configurabilité à la section 5.2. Les résultats sur *OSGi* sont donnés dans la section 5.3. La section 5.4 traite des résultats obtenus sur l'implémentation du modèle *TAC*. Les résultats sur l'assistant à la préparation d'une recette de cuisine sont discutés dans la section 5.5. Les fichiers de configurations des deux prototypes sont fournis dans la section 5.6. Des interfaces modulables sont proposées dans la section 5.7.

5.1 Lecture sur la méthodologie

Une fois que le choix du modèle TAC comme cœur de mon projet fut arrêté, il a fallu développer une structure propre pour respecter les exigences de l'assistance cognitive. Le cahier des charges était simple :

1. Fort potentiel de configuration, car les profils et les besoins des personnes atteintes de déficits cognitifs sont très variés ;
2. Ouverture pour une architecture OSGi, car DOMUS adoptera cette architecture pour les futurs développements ;
3. Respect du modèle TAC à la lettre, ce que conseille l'orienté objet.

Le choix de réaliser une application virtuelle découle du projet initial qui était de réaliser une table interactive pour l'exposition permanente du Centre de Sciences de Montréal. Pendant la conception du stand, plusieurs propositions ont émergé dont la

manipulation virtuelle de recettes de cuisine. La construction de l'application virtuelle a permis la mise au point de la couche TAC complète.

La première phase a été de concevoir et tester à l'aide de tests unitaires la couche TAC. Les tests unitaires sont basés sur le modèle TAC du projet Designer's Outpost. Ensuite reacTIVision, découvert lors du développement de ce projet, m'a donné une base solide pour construire et tester le TAC concrètement. Le lien entre la couche matérielle simulée reacTIVision et la couche intermédiaire TAC s'est clarifié alors. Enfin, la construction de l'application finale n'a pu être finalisée comme il avait été initialement prévu.

À l'origine, en plus de la modélisation de la recette, chaque contenant devait décrire son contenu. La structure des données du contenant aurait été comparée à un modèle étalon pour déterminer si la recette était finalisée ou pas à l'aide d'une fonction de similarité. L'assistance aurait été capable de déterminer les actions à effectuer pour avancer. Finalement, la structure est devenue trop complexe. L'essentiel en a toutefois été conservé, ainsi un contenant est capable de décrire son contenu. Par contre, l'assistance est devenue plus faible, car le contrôle de la procédure est effectué grâce à un réseau de Petri. Cela a permis de commencer le développement d'une application qui utilise bel et bien une cuisine réelle.

Au plan de l'organisation du code, le projet complet était contenu dans un seul projet Eclipse, ce qui n'offrait aucune visibilité pour savoir si l'application était bien découplée en modules. Effectivement, quelques classes faisaient des raccourcis qui n'auraient pas dû exister. Une fois le programme totalement décortiqué en modules, j'ai tenté un passage sous OSGi. Des résultats intéressants ont été obtenus et certains liens entre les modules ont dû être changés. J'en parlerai dans la section dédiée à OSGi de ce chapitre.

Enfin, la construction de la deuxième application a permis de valider la méthode de développement et d'uniformiser certain éléments, comme les liens entre la couche matérielle et la couche intermédiaire TAC ou le code de l'assistant commun aux deux prototypes. Je pensais par exemple utiliser les mêmes variables pour les deux applications, mais les actions étant complètement différentes, cela ne pouvait pas cohabiter, par exemple le prototype virtuel propose des actions de haut niveau (chauffer, verser, etc.) alors que le prototype réel contrôle le déplacement des objets. Du coup, un module commun aux deux applications définit seulement les liens entre le réseau de Petri et l'affichage ambiant de l'aide. Chaque application est un module qui lie toutes les couches à des classes finales définies dans le module. Ainsi, le développement des deux applications principales est terminé.

En parallèle des interfaces modulables ont été conçues et certaines sont utilisées par l'application réelle. Ces résultats seront vus dans la section 5.2.

5.2 Architecture logicielle et configurabilité

Les modules sont simples et configurables grâce à *XML*. La dernière couche doit être programmée pour lier les couches entre elles, ce qui nécessite des capacités de programmation et repose sur la configuration en XML de l'application. Les programmes personnalisables restent malgré tout difficiles à configurer. Par ailleurs, il est difficile de maintenir ce type d'applications à cause des liens de dépendance entre les diverses technologies et services utilisés. C'est pourquoi le fonctionnement du système et les interrelations entre les modules ont été clairement expliquées au chapitre 2.

Aspect essentiel de mon travail, pour pouvoir configurer les structures, l'introspection permet de sélectionner les implémentations utilisées. Or l'introspection ne peut fonctionner lorsque les classes sont manquantes, ce qui peut se révéler hautement problématique dans un contexte d'assistance cognitive.

Je pense que les fichiers de configurations font partie du développement de l'application. Si un échec est produit alors, le programme est mal développé. Un outil de configuration pourrait tester la validité et la cohérence des fichiers de configurations.

5.3 OSGi

La réalisation sous OSGi a été plus compliquée que prévu, même si les modules étaient bien découplés et l'application fonctionnelle. Les problèmes proviennent des dépendances des librairies.

Premièrement l'initialisation *XML* se basait sur les librairies utilitaires de DOMUS qui elles-mêmes utilisaient *Xerces* entre autres. Bien qu'il soit possible d'inclure des librairies statiques *JAR* dans un *Bundle*, l'application ne fonctionnait pas encore. Des exceptions *ClassNotFound* à la première exécution et une erreur à la deuxième étaient lancées. L'introspection utilisée lors de l'initialisation *XML* causait ces échecs. J'ai changé toutes les lectures des fichiers *XML* vers une structure *kXML* [1]. L'initialisation *XML* fonctionnait suite à une adaptation de la librairie *kXML*.

[1] Boîte à outil pour la lecture des fichiers XML. Site web : kxml.sourceforge.net/kxml2/

Deuxièmement ma propre librairie générait des *ClassCastExceptions*. Ces erreurs provenaient du fait que le *ClassLoader* d'un *Bundle* ne peut pas charger des classes d'un module dont il ne dépend pas. Car un *ClassLoader* ne peut déterminer de quel module charger en premier lieu. Une solution toute simple a été trouvée. La couche TAC est dotée d'un singleton qui est chargé de stocker les classes utilisées par l'introspection. Ainsi avant l'initialisation de la couche TAC, la dernière couche enregistre les classes utilisées par l'initialisation de la couche TAC.

Troisièmement, les structures du réseau de Petri et de l'affichage ambiant ont eu le même problème. La même solution a été appliquée à chaque fois que l'introspection doit charger des classes hors de son champ de vision.

Je ne suis pas allé plus loin avec OSGi. Par exemple, je n'ai pas rendu orientée service mon application, ni même utilisé les services offerts. Je veux que mon application fonctionne avec ou sans OSGi. OSGi offre une grande ouverture, mais la structure limite l'utilisation massive de l'introspection, nécessaire pour construire des frameworks configurables par fichiers.

5.4 Couche TAC

La réalisation de la couche est un succès. J'ai conçu des tests unitaires (*JUnit*) sur la base de la modélisation du projet DesignerOutpost [32]. J'ai des classes spécifiques pour lier la couche afin de simuler la vraie utilisation et je teste que les méthodes des variables sont exécutées lorsque les objets sont associés aux bonnes contraintes et lorsque les bonnes manipulations ont lieu.

TAC permet de modéliser les interactions tangibles de type *Token+Constraint*. Je pense qu'il est possible de modéliser d'autres interactions avec ce modèle.

La structure TAC permet de contrôler les actions physiques. Les manipulations sur les variables sont autorisées si et seulement si certaines conditions sont remplies, c'est-à-dire que l'objet satisfait toutes les contraintes définies. L'invocation de méthode sur la variable associée à un TAC permet de la modifier.

5.5 Assistant à la préparation de recettes de cuisine

Mon application actuelle vérifie que les bonnes actions sont faites. Il n'existe pas de représentation de la recette.

Je pense que le choix d'un bon assistant n'est pas facile. Mon travail est suffisamment structuré pour permettre un changement d'implémentation. Seul TAC impose la modification d'objets avec l'invocation de méthodes. Je ne définis pas de structure pour la couche du haut.

Ma proposition est un assistant qui contrôle l'invocation de méthodes et une modélisation des contenus physiques. L'assistant que je propose n'est pas complet, mais de futurs travaux pourront apporter de meilleures réponses. Par contre, le fonctionnement est bien déterminé avec les réseaux de Petri, ce qui a l'avantage de bien montrer comment l'application fonctionne.

Il faut décrire les manipulations physiques dans TAC, puis décrire les bonnes manipulations pour chaque recette, ainsi que les erreurs et la description de récupération. La description complète de tout cela est complexe. Je pense qu'il faut réfléchir à une autre solution, notamment avec un outil de configuration.

Les manipulations que j'ai décrites en TAC semblent complètes, mais je ne propose qu'une seule recette. RecipeTable propose une interface tangible pour sélectionner une recette suivant les ingrédients présents sur la table interactive. Il faudrait proposer une solution similaire et commencer la description de plus de recettes.

Les manipulations du prototype virtuel sont présentées dans la sous-section 5.5.1,. Les manipulations du prototype réel est présentées dans la sous-section 5.5.2.

5.5.1 Les manipulations du prototype virtuel

Il y a cinq étapes dans le scénario proposé, qui est de cuire du riz. La première étape consiste à verser de l'eau dans la casserole (fig. 5.1). Ensuite, il faut faire bouillir l'eau (fig. 5.2). Le riz est alors versé dans l'eau bouillante (fig. 5.3). Le riz sera alors cuit (fig. 5.4). Pour finir, les riz est passé (fig. 5.5).

La figure 5.1 présente les manipulations pour verser de l'eau dans la casserole. Il y a quatre étapes, qui consiste à :

1. déposer le jeton "casserole" dans la contrainte "contenant",
2. déposer le jeton "eau" dans la contrainte "contenu à verser",
3. de tourner le jeton "eau" dans la contrainte "contenu à verser" jusqu'à obtenir la quantité souhaitée,
4. de sortir le jeton "eau" de la contrainte "contenu à verser", afin de verser la quantité saisie.

La figure 5.2 présente les manipulations pour faire bouillir l'eau. Il y a trois étapes, qui consiste à :

1. déposer le jeton "casserole" dans la contrainte "chauffer",
2. de tourner le jeton "casserole" dans la contrainte "chauffer" jusqu'à obtenir de l'eau bouillante,
3. de sortir le jeton "casserole" de la contrainte "chauffer", afin de saisir la température souhaitée.

La figure 5.3 présente les manipulations pour verser du riz dans l'eau bouillante. Il y a quatre étapes, qui consiste à :

1. déposer le jeton "casserole" dans la contrainte "contenant",
2. déposer le jeton "riz" dans la contrainte "contenu à verser",
3. de tourner le jeton "riz" dans la contrainte "contenu à verser" jusqu'à obtenir la quantité souhaitée,
4. de sortir le jeton "riz" de la contrainte "contenu à verser", afin de verser la quantité saisie.

La figure 5.4 présente les manipulations pour faire cuire le riz. Il y a trois étapes, qui consiste à :

1. déposer le jeton "casserole" dans la contrainte "chauffer",
2. de tourner le jeton "casserole" dans la contrainte "chauffer" jusqu'à obtenir "riz cuit",
3. de sortir le jeton "casserole" de la contrainte "chauffer", afin de saisir la température souhaitée.

La figure 5.5 présente les manipulations pour passer le riz. Il y a une étape, qui consiste à déposer le jeton "casserole" dans la contrainte "passer", le riz sera alors passé.

5.5.2 Les manipulations du prototype réel

Il y a cinq étapes dans le scénario proposé, qui est de cuire du riz. La première étape consiste à verser de l'eau dans la casserole (fig. 5.6, 5.7). Ensuite, il faut faire bouillir l'eau (fig. 5.8). Le riz est alors versé dans l'eau bouillante (fig. 5.9, 5.10, 5.11). Le riz sera alors cuit (fig. 5.8). Pour finir, le riz est passé (fig. 5.12, 5.13).

Les figures 5.6 et 5.7 présentent les manipulations pour verser de l'eau dans la casserole. Il y a six étapes, qui consiste à :

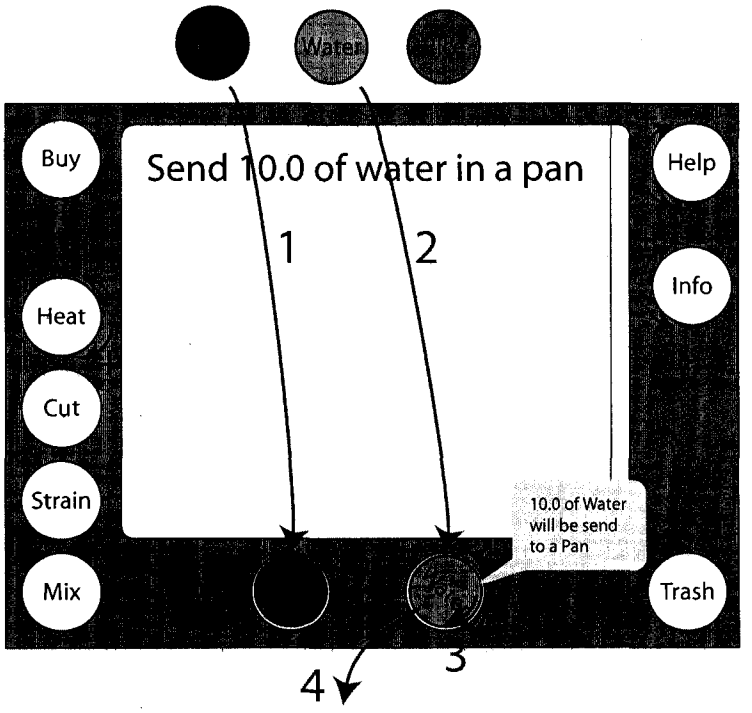

FIGURE 5.1 – Verser de l'eau dans la casserole

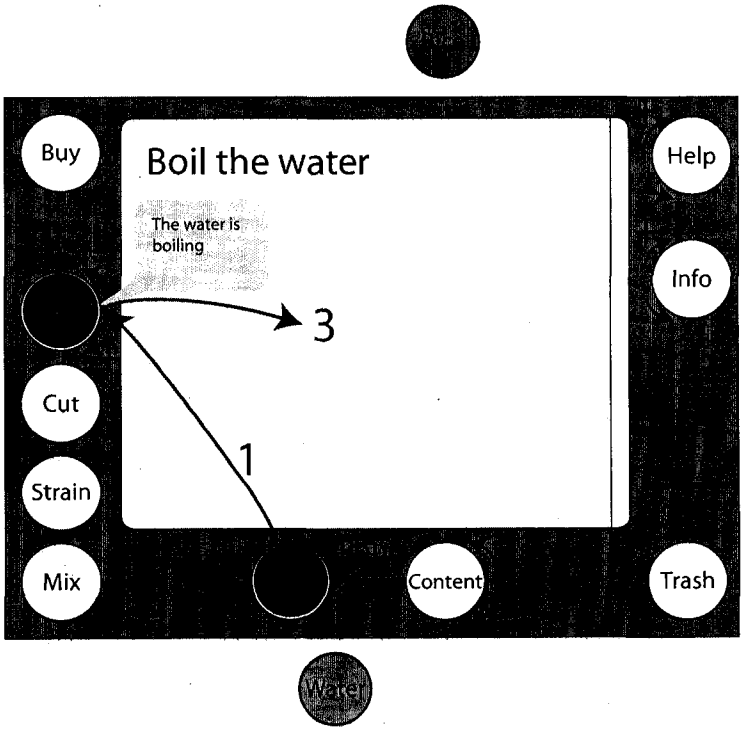

FIGURE 5.2 – Faire bouillir l'eau

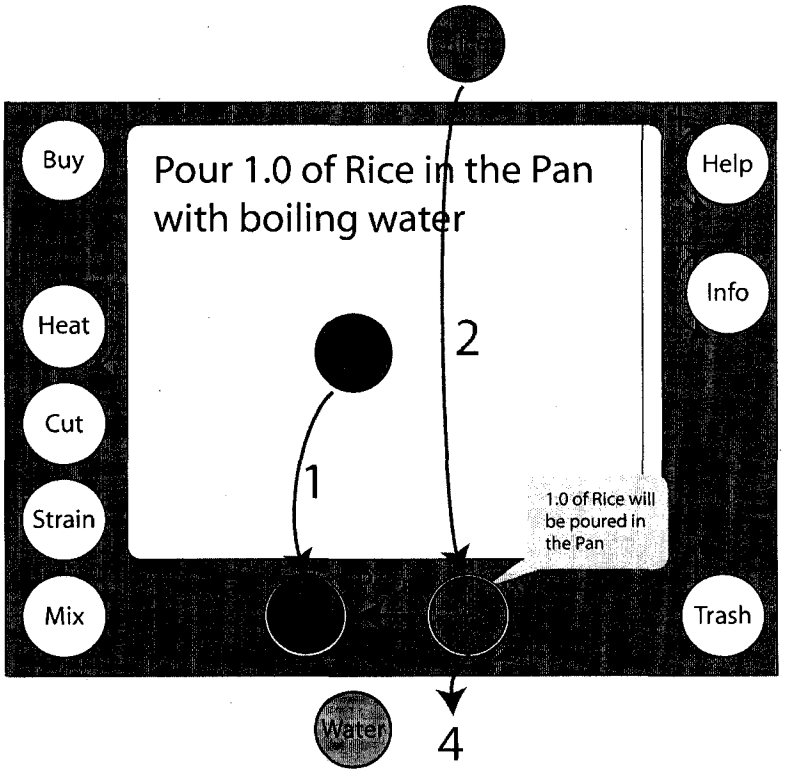

FIGURE 5.3 – Verser le riz dans l'eau bouillante

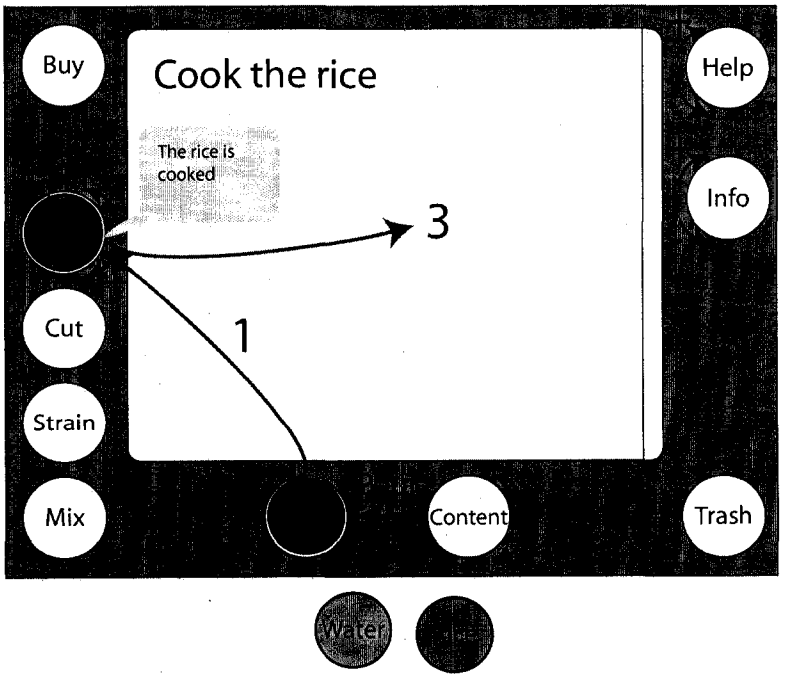

FIGURE 5.4 – Cuire le riz

1. ouvrir le placard, action détectée par un capteur diffus d'ouverture de porte,
2. sortir la casserole du placard, action détectée par RFID et transmise au TAC,
3. fermer le placard, action détectée par un capteur diffus de fermeture de porte,
4. poser la casserole dans l'évier, action détectée par RFID et transmise au TAC,
5. ouvrir le robinet, action détectée par les débitmètres,
6. fermer le robinet, action détectée par les débitmètres.

La figure 5.8 présente les manipulations pour faire bouillir l'eau. Il y a deux étapes, qui consiste à :

1. placer la casserole sur une plaque chauffante, action détectée par RFID et transmise au TAC,
2. allumer la plaque, action détectée par un capteur sur l'interrupteur de la plaque.

Les figures 5.9, 5.10 et 5.11 présentent les manipulations pour verser le riz dans l'eau bouillante. Il y a cinq étapes, qui consiste à :

1. ouvrir le placard, action détectée par un capteur diffus d'ouverture de porte,
2. sortir le riz et le verre d'eau du placard, actions détectées par RFID et transmises au TAC,
3. fermer le placard, action détectée par un capteur diffus de fermeture de porte,
4. verser le riz dans le verre sur le plan de travail, action détectée par RFID et transmise au TAC,
5. verser le riz du verre dans l'eau bouillante, action détectée par RFID et transmise au TAC,

Les figures 5.12 et 5.13 présentent les manipulations pour passer le riz. Il y a sept étapes, qui consiste à :

1. ouvrir le placard, action détectée par un capteur diffus d'ouverture de porte,
2. sortir la passoire du placard, action détectée par RFID et transmise au TAC,
3. fermer le placard, action détectée par un capteur diffus de fermeture de porte,
4. placer la passoire dans l'évier, action détectée par RFID et transmise au TAC,
5. fermer le feu de la plaque, action détectée par un capteur sur l'interrupteur de la plaque,
6. sortir la casserole du feu, action détectée par RFID et transmise au TAC,
7. verser le riz dans la passoire, action détectée par RFID et transmise au TAC.

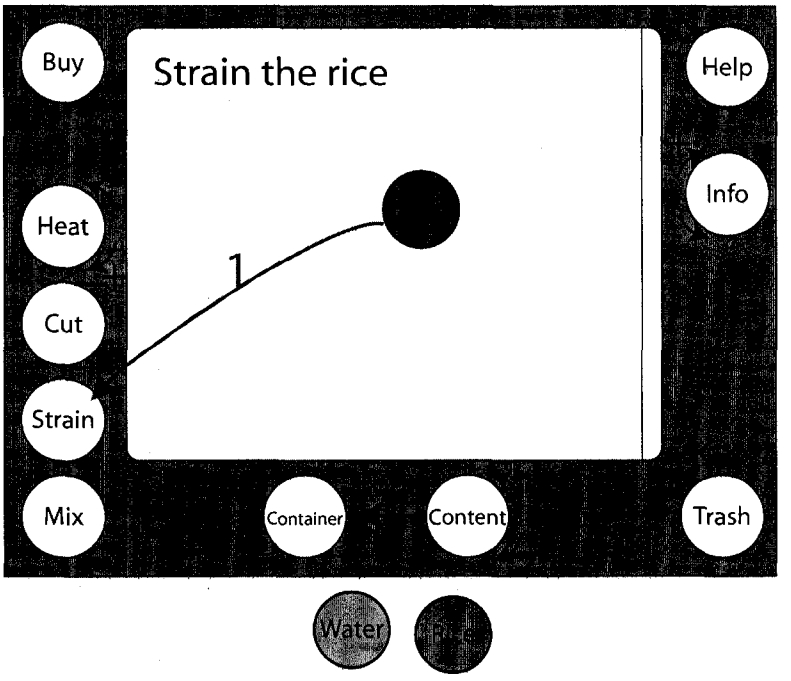

FIGURE 5.5 – Passer le riz

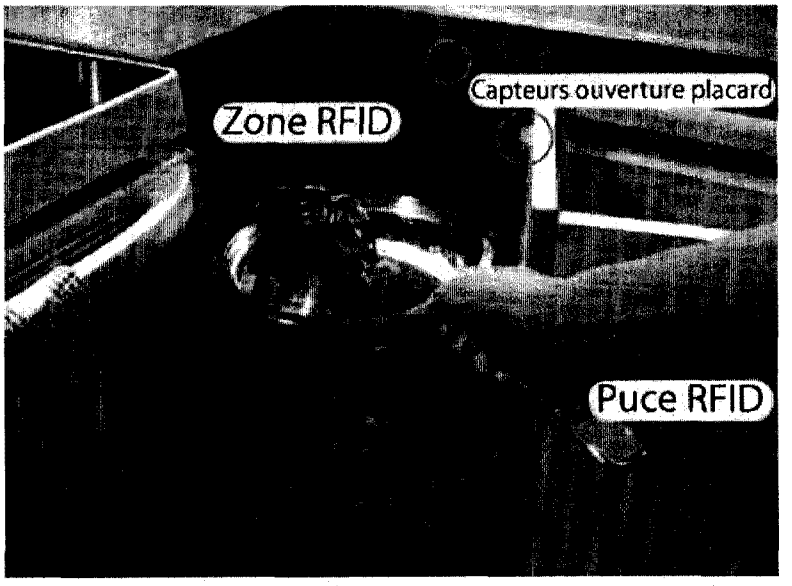

FIGURE 5.6 – Sortir la casserole du placard

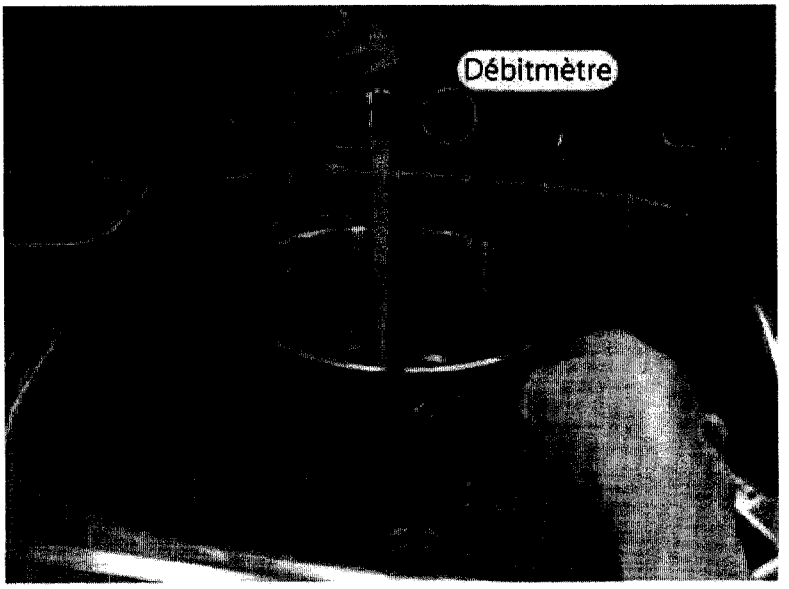

FIGURE 5.7 – Verser de l'eau dans la casserole

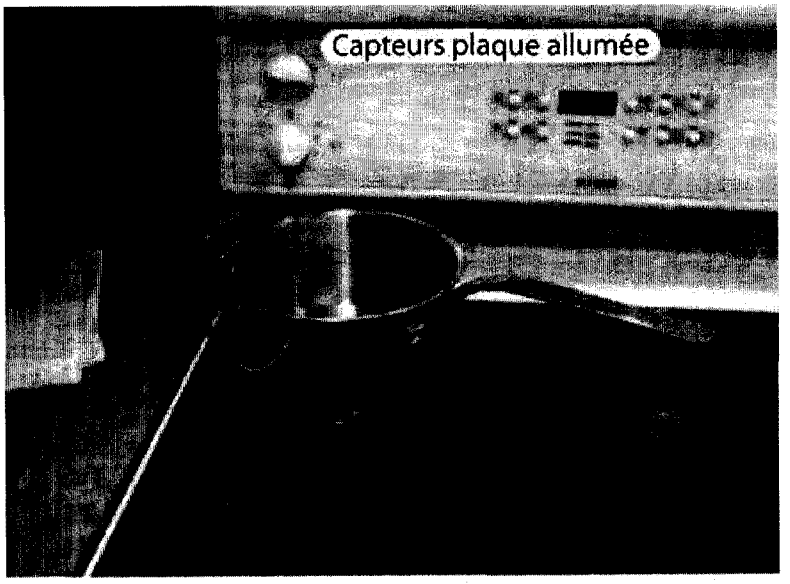

FIGURE 5.8 – Faire bouillir l'eau ou cuire le riz

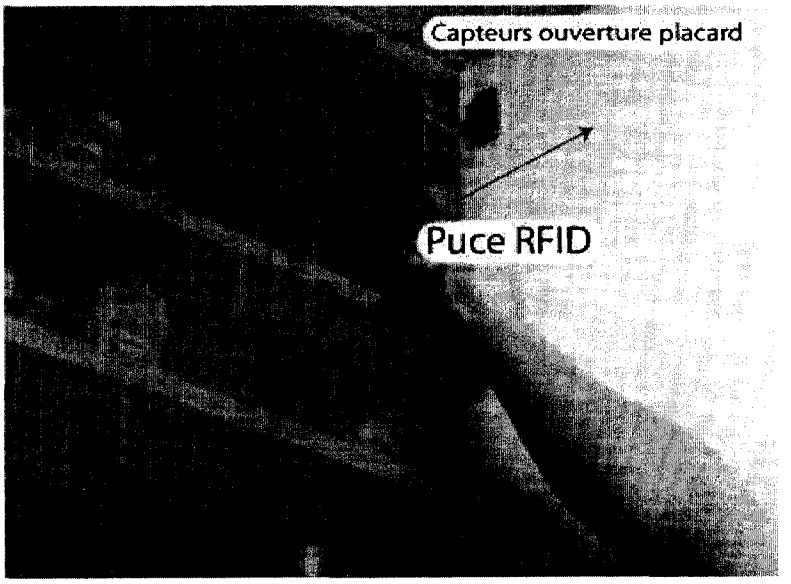

FIGURE 5.9 – Sortir le riz et un verre d'eau du placard

FIGURE 5.10 – Verser le riz dans le verre d'eau

5.6 Configurations des applications

Les fichiers de configurations sont multiples et différents suivant l'application. Les exemples sont fournis dans l'annexe B et C pour l'application virtuelle et réelle respectivement.

5.6.1 Application avec les manipulations virtuelles

L'application virtuelle utilise *reacTIVision* comme couche matérielle et un affichage graphique projeté.

L'annexe B contient tous les fichiers de configuration utilisés par l'application virtuelle.

5.6.2 Application avec les manipulations réelles

L'application réelle utilise *Symbol RFID* et des capteurs diffus comme couche matérielle et un affichage ambiant avec un *Phidget Text LCD* et *Phidget Servo Motor.*

L'annexe C contient tous les fichiers de configuration utilisés par l'application réelle.

5.7 Interfaces modulables

Quelques interfaces ont été réalisées : l'afficheur de texte LCD et l'afficheur à aiguilles. Elles ont été utilisées dans l'assistant. D'autres interfaces ambiantes existent déjà, notamment les lumières qui sont contrôlées par réseau électrique et les diodes électroluminescentes sur les placards.

L'afficheur texte affiche les instructions pour accomplir la tâche de manière textuelle.

L'afficheur à aiguilles présente l'accomplissement et l'étape courante. Le sens des aiguilles est modélisé dans l'application, il faut imprimer un cadran représentant le sens souhaité.

Je pense, ainsi que d'autres personnes qui travaillent dans les interfaces ambiantes, que l'affichage d'images et de vidéos doit passer par un cadre photo interactif. En effet, ces cadres interactifs sont de petits écrans *LCD* décoratifs qui ne présentent aucune interface d'entrée humaine (pour les modèles les plus simples). Une connexion sans fil permet au système de mettre à jour le contenu affiché. Je n'ai pas réalisé cet affichage, mais c'est une solution à envisager fortement. Les ordinateurs portatifs sont utilisés pour le moment.

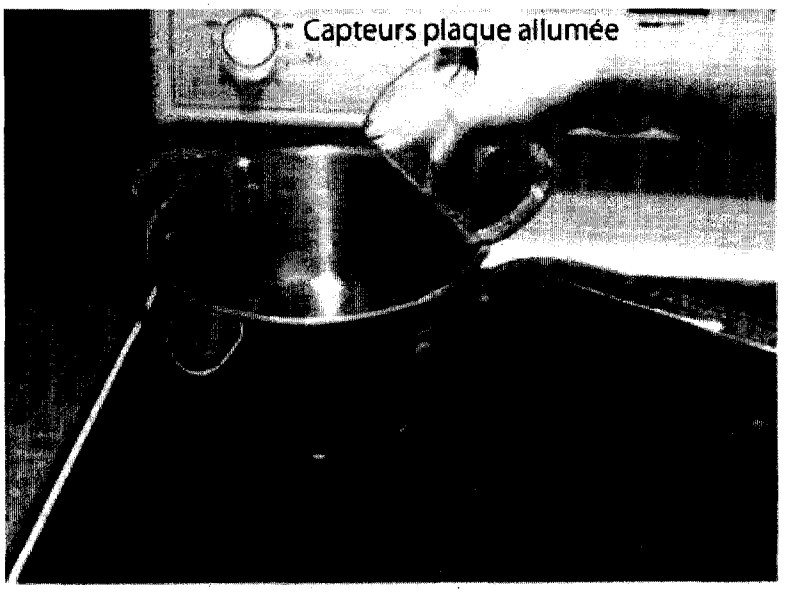

FIGURE 5.11 – Verser le verre d'eau plein de riz dans la casserole

FIGURE 5.12 – Sortir la passoire du placard

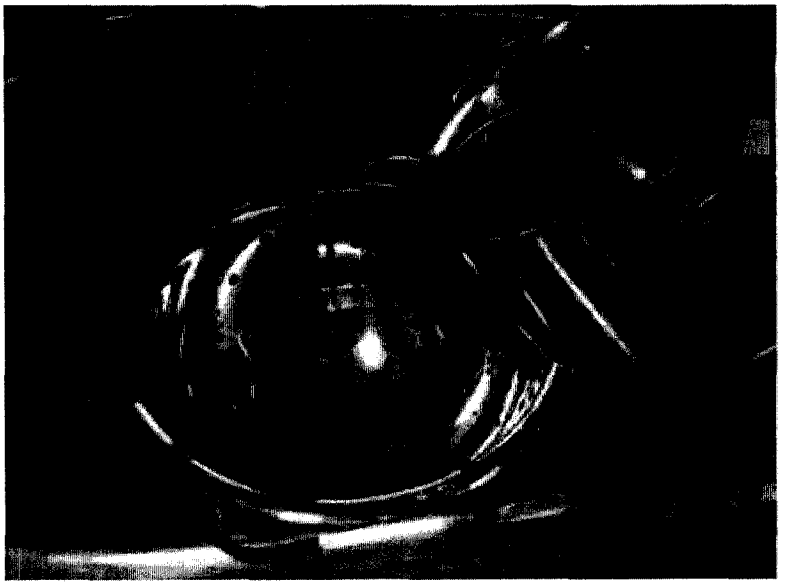

FIGURE 5.13 – Mettre la passoire dans l'évier et passer le riz à traver la passoire

Conclusion générale

Ce mémoire ouvre un thème nouveau au laboratoire DOMUS qui nécessite encore beaucoup de travail et d'approfondissement. Les interfaces émergentes, dont la conception fait encore débat, sont appliquées à l'assistance aux personnes ayant des déficits cognitifs. Un assistant cognitif basé sur une architecture informatique utilise des claviers, des écrans et des souris. Notre travail a supprimé ces interfaces, qui ne sont pas adaptées pour les personnes ayant des déficiences cognitives, en les remplaçant par des interfaces tangibles. Nous avons cherché quelles interfaces tangibles peuvent être appliquées à l'assistance, et cela, en respectant certaines recommandations issues de l'assistance cognitive et de l'éthique.

La personnalisation de l'assistant est une des règles les plus importantes à respecter pour la réalisation d'une assistane cognitive adéquate. Mon travail a apporté des éléments de réponse dans ce domaine avec la conception d'une architecture complètement configurable à l'aide de fichiers *XML*. Aussi les structures élémentaires permettent un assemblage à la demande. Par contre, pour le moment seul un développeur peut accomplir ce travail. La famille ou le médecin n'ont pas vocation à produire un travail de développement et de configuration aussi complexe.

Il faut que l'assistant soit conçu à dimension humaine, c'est-à-dire qu'il soit respectueux et compréhensible par la personne. Mon travail n'a pas permis d'apporter de réponses à ce problème. Il aurait fallu réaliser une étude en milieu réel pour le déterminer. Toutefois, certaines règles doivent primer. Premièrement, le système n'utilisera pas de caméra. Ensuite, une interface d'entrée, conçue pour la personne, permettra à celle-ci de prendre le contrôle de la machine et non l'inverse.

Nous avons appliqué le modèle *Token and Constraint* (TAC), pour modéliser les interactions physiques et virtuelles. Des variables virtuelles sont associées aux objets. Les interactions de l'homme sur ces objets impliquent des opérations sur les variables associées.

Deux prototypes servant à assister la personne à la préparation de recettes de cuisine ont été conçus avec comme cœur le modèle *TAC*. Le premier prototype a permis de faire une preuve de concept quant à la capacité des interfaces tangibles à remplacer les claviers, les souris et les écrans, dans le cadre de l'assistance cognitive aux personnes ayant des troubles cognitifs. Ce prototype consiste en une table interactive où des objets (cubes, palets) représentant des ingrédients et des contenants. Ces objets sont manipulés suivant les fonctions affichées sur la table (verser, chauffer, mélanger, etc.). La modélisation de ces opérations de cuisine est intéressante et pourrait servir de base pour la base d'un nouveau modèle d'assistance. Le second prototype a permis de réaliser des interfaces tangibles dans une vraie cuisine. Les objets sont plus concrets, car nous utilisons la casserole, la passoire, le riz et le verre d'eau, au lieu de cubes et de palets représentant des objets de même nature. Ce dernier prototype montre qu'il est donc possible d'assister une personne ayant des déficiences cognitives à l'aide des interfaces tangibles, et ce, sans écrans, claviers et souris.

Des problèmes restent toutefois à résoudre sur le plan matériel. En effet, la conception d'une interface tangible diffuse dans l'environnement nécessite l'identification et la localisation des objets. Les systèmes d'identification et de localisation des objets ne sont pas encore suffisamment précis et au point. Ces technologies devraient offrir de bien meilleurs résultats dans un avenir rapproché.

Mon travail a ainsi permis d'explorer les interfaces tangibles appliquées à l'assistance cognitive. Il permet désormais de travailler sur des thématiques plus précises. En voici quelques-unes : les interfaces ambiantes, l'application d'une assistance plus évoluée, les cartes iconiques interactives.

Un programme de configuration visuel est nécessaire afin de rendre plus simple la conception de l'assistant cognitif. Il faudrait en particulier produire des modules configurés et assemblés visuellement. Ce programme serait capable de décrire les besoins matériels et de générer le code pour s'interfacer avec le monde réel. Le médecin ou la famille pourront ainsi configurer l'assistant.

Le domaine de l'assistance cognitive manque de modèles définis. Certaines règles de conduite pour la conception des interfaces et de l'assistance cognitive ont été énoncées dans ce mémoire. Le modèle *Token and Constraint* a été proposé comme moyen de décrire les interactions tangibles avec les objets. Le laboratoire DOMUS a besoin d'autres modèles et ontologies, comme pour l'environnement ou le profil utilisateur.

Enfin, mes résultats devraient permettre d'avoir une meilleure vision sur le dévelop-

pement d'un assistant configurable. Un programme de configuration et un modèle unifié pourraient grandement aider à l'avancement du projet dans son ensemble. Je crois que ces étapes peuvent être désormais engagées, parce que la compréhension générale du problème est maintenant suffisamment mature. J'espère que ce travail et mes remarques serviront à appuyer les choix techniques dans le futur.

Annexe A

Documentation XML

A.1 Notation

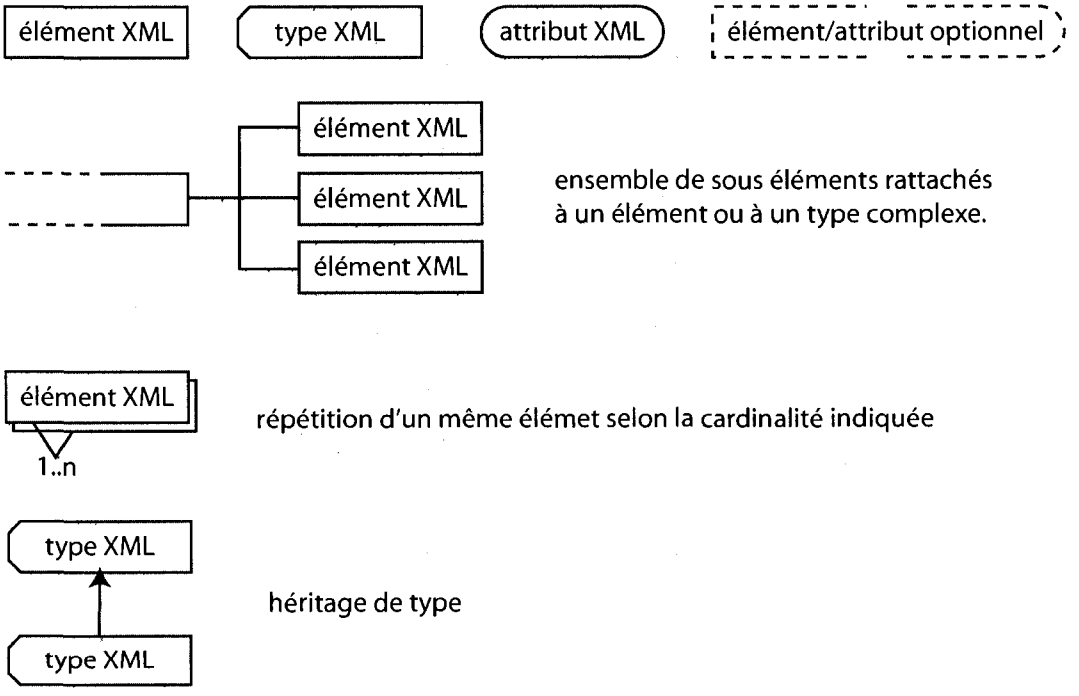

FIGURE A.1 – Notation visuelle XML schéma

A.2 Structure fichier XML TAC

```
<?xml version="1.0" encoding="UTF-8"?>
<tui xmlns:xdo="http://intranet.domus.usherbrooke.ca"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://intranet.domus.usherbrooke.ca
 http://intranet.domus.usherbrooke.ca/fichiers/schemas/tui/modeltac.xsd">
        <tacs>
                <tac>
                        <name>1</name>
                        <token>Task</token>
                        <constraints>
                                <constraint>Help</constraint>
                                ...
                        </constraints>
                        <variable>path.to.package.Class</variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen/>
                                        <method>displayHelp</method>
                                </action>
                                ...
                        </actions>
                </tac>
                ...
        </tacs>
        <constraintsinstances>
                <constraintinstance>
                        <type>Help</type>
                        <name>Aide</name>
                        <listen>path.to.package.Class</listen>
                </constraintinstance>
                ...
        </constraintsinstances>
        <pyfos>
                <pyfo>
                        <type>Task</type>
                        <name>Tache a faire</name>
                        <variable>
```

```
                        <factory>false</factory>
                        <clazz>path.to.package.Class</clazz>
                        <method/>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Cuire du riz</value>
                                </argument>
                                ...
                        </arguments>
                </variable>
        </pyfo>
        ...
    </pyfos>
</tui>
```

Annexe B

Fichiers de configuration pour l'application virtuelle

B.1 reacTIVision

```
<?xml version="1.0" encoding="UTF-8"?>
<tuio>
        <monitor>
                <clazz>ca.usherbrooke.domus.reactivision.monitor.VgaMonitor</clazz>
                <name>TuioTest</name>
        </monitor>
        <zones>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Acheter</name>
                        <x>0.2</x>
                        <y>0.2</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Chauffer</name>
```

```
                        <x>0.2</x>
                        <y>0.35</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Passoire</name>
                        <x>0.2</x>
                        <y>0.475</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Couper</name>
                        <x>0.2</x>
                        <y>0.6</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Mixer</name>
                        <x>0.2</x>
                        <y>0.725</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Contenu</name>
                        <x>0.4</x>
```

```
                        <y>0.5</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Contenant</name>
                        <x>0.6</x>
                        <y>0.5</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Aide</name>
                        <x>0.80</x>
                        <y>0.2</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Info</name>
                        <x>0.80</x>
                        <y>0.35</y>
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
                <zone>
                        <classname>
ca.usherbrooke.domus.reactivision.zone.ZoneRectangle
                        </classname>
                        <name>Jeter</name>
                        <x>0.80</x>
                        <y>0.70</y>
```

```
                        <width>0.075</width>
                        <height>0.1</height>
                </zone>
        </zones>
</tuio>
```

B.2 Liaison reacTIVision et TAC

```
<?xml version="1.0" encoding="UTF-8"?>
<fiducials>
        <fiducial>
                <id>0</id>
                <pyfoname>Pan</pyfoname>
        </fiducial>
        <fiducial>
                <id>12</id>
                <pyfoname>Rice</pyfoname>
        </fiducial>
        <fiducial>
                <id>32</id>
                <pyfoname>Water</pyfoname>
        </fiducial>
        <fiducial>
                <id>36</id>
                <pyfoname>Cook rice</pyfoname>
        </fiducial>
</fiducials>
```

B.3 TAC

```
<?xml version="1.0" encoding="UTF-8"?>
<tui xmlns:xdo="http://intranet.domus.usherbrooke.ca"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://intranet.domus.usherbrooke.ca
http://intranet.domus.usherbrooke.ca/fichiers/schemas/tui/modeltac.xsd">
        <tacs>
                <tac>
                        <name>1</name>
                        <token>Task</token>
```

```
                        <constraints>
                                <constraint>Help</constraint>
                        </constraints>
                        <variable>
                                ca.usherbrooke.domus.kitchenaid.variable.Task
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>displayHelp</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>hideHelp</method>
                                </action>
                                <action>
                                        <name>scrollUp</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.TaskHelpScrollUp
                                        </listen>
                                        <method>scrollUp</method>
                                </action>
                                <action>
                                        <name>scrollDown</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.TaskHelpScrollDown
                                        </listen>
                                        <method>scrollDown</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>2</name>
                        <token>Task</token>
                        <constraints>
```

```
                        <constraint>Information</constraint>
                    </constraints>
                    <variable>
                        ca.usherbrooke.domus.kitchenaid.variable.Task
                    </variable>
                    <actions>
                        <action>
                            <name>add</name>
                            <automatic>true</automatic>
                            <listen />
                            <method>displayInformation</method>
                        </action>
                        <action>
                            <name>remove</name>
                            <automatic>true</automatic>
                            <listen />
                            <method>hideInformation</method>
                        </action>
                        <action>
                            <name>scrollUp</name>
                            <automatic>false</automatic>
                            <listen>
ca.usherbrooke.domus.kitchenaid.hardware.TaskInformationScrollUp
                            </listen>
                            <method>scrollUp</method>
                        </action>
                        <action>
                            <name>scrollDown</name>
                            <automatic>false</automatic>
                            <listen>
ca.usherbrooke.domus.kitchenaid.hardware.TaskInformationScrollDown
                            </listen>
                            <method>scrollDown</method>
                        </action>
                    </actions>
                </tac>
                <tac>
                    <name>3</name>
                    <token>Task</token>
                    <constraints>
                        <constraint>Trash</constraint>
```

```
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.variable.Task
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>deleteTask</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>4</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Information</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>displayInformation</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>hideInformation</method>
                                </action>
                                <action>
                                        <name>scrollUp</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentInformationScrollUp
                                        </listen>
```

```
                                        <method>scrollUp</method>
                                </action>
                                <action>
                                        <name>scrollDown</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentInformationScrollDown
                                        </listen>
                                        <method>scrollDown</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>5</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Pour-Destination</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>prepareToReceiveContent</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>deletePreparedToReceiveState</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>6</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Pour-Source</constraint>
```

```
                        </constraints>
                        <variable>
                                ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>prepareToSendContent</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>sendContent</method>
                                </action>
                                <action>
                                        <name>scrollUp</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentVerseSourceScrollUp
                                        </listen>
                                        <method>scrollUp</method>
                                </action>
                                <action>
                                        <name>scrollDown</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentVerseSourceScrollDown
                                        </listen>
                                        <method>scrollDown</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>7</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Help</constraint>
                        </constraints>
```

```
                        <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>displayHelp</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>hideHelp</method>
                                </action>
                                <action>
                                        <name>scrollUp</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentHelpScrollUp
                                        </listen>
                                        <method>scrollUp</method>
                                </action>
                                <action>
                                        <name>scrollDown</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentHelpScrollDown
                                        </listen>
                                        <method>scrollDown</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>8</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Trash</constraint>
                        </constraints>
                        <variable>
```

```
ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>delete</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>9</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Buy</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>buy</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>10</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Mix</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
```

```
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>mix</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>11</name>
                        <token>Container</token>
                        <constraints>
                                <constraint>Heat</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>heat</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>stopHeating</method>
                                </action>
                                <action>
                                        <name>scrollUp</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentHeatScrollUp
                                        </listen>
                                        <method>scrollUp</method>
                                </action>
                                <action>
                                        <name>scrollDown</name>
                                        <automatic>false</automatic>
                                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.ContentHeatScrollDown
```

```
                                </listen>
                                <method>scrollDown</method>
                        </action>
                </actions>
        </tac>
        <tac>
                <name>12</name>
                <token>Container</token>
                <constraints>
                        <constraint>Cut</constraint>
                </constraints>
                <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                </variable>
                <actions>
                        <action>
                                <name>add</name>
                                <automatic>true</automatic>
                                <listen />
                                <method>cut</method>
                        </action>
                </actions>
        </tac>
        <tac>
                <name>13</name>
                <token>Container</token>
                <constraints>
                        <constraint>Strain</constraint>
                </constraints>
                <variable>
ca.usherbrooke.domus.kitchenaid.variable.Content
                </variable>
                <actions>
                        <action>
                                <name>add</name>
                                <automatic>true</automatic>
                                <listen />
                                <method>strain</method>
                        </action>
                </actions>
        </tac>
```

```
        </tacs>
        <constraintsinstances>
                <constraintinstance>
                        <type>Help</type>
                        <name>Aide</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.Help
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Information</type>
                        <name>Informations</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.Information
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Trash</type>
                        <name>Jeter</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.Drop
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Buy</type>
                        <name>Acheter</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.Buy
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Mix</type>
                        <name>Mélanger</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.Mix
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Heat</type>
                        <name>Chauffer</name>
                        <listen>
```

```
ca.usherbrooke.domus.kitchenaid.hardware.Heat
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Cut</type>
                        <name>Couper</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.Cut
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Pour-Source</type>
                        <name>Contenu à verser</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.PourSource
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Pour-Destination</type>
                        <name>Contenant</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.PourDestination
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Strain</type>
                        <name>Passoire</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.hardware.Strain
                        </listen>
                </constraintinstance>
        </constraintsinstances>
        <pyfos>
                <pyfo>
                        <type>Task</type>
                        <name>Cook rice</name>
                        <variable>
                                <factory>true</factory>
                                <clazz>
ca.usherbrooke.domus.kitchenaid.variable.TaskFactory
                                </clazz>
```

```
                        <method>createTask</method>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Cook rice</value>
                                </argument>
                        </arguments>
                </variable>
        </pyfo>
        <pyfo>
                <type>Container</type>
                <name>Pan</name>
                <variable>
                        <factory>true</factory>
                        <clazz>
ca.usherbrooke.domus.kitchenaid.variable.ContentFactory
                        </clazz>
                        <method>createEmptyContent</method>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Content of the pan</value>
                                </argument>
                        </arguments>
                </variable>
        </pyfo>
        <pyfo>
                <type>Container</type>
                <name>Water</name>
                <variable>
                        <factory>true</factory>
                        <clazz>
ca.usherbrooke.domus.kitchenaid.variable.ContentFactory
                        </clazz>
                        <method>createProduct</method>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Water</value>
                                </argument>
                                <argument>
```

```
                                        <order>1</order>
                                        <value>1000</value>
                                </argument>
                        </arguments>
                </variable>
        </pyfo>
        <pyfo>
                <type>Container</type>
                <name>Rice</name>
                <variable>
                        <factory>true</factory>
                        <clazz>
ca.usherbrooke.domus.kitchenaid.variable.ContentFactory
                        </clazz>
                        <method>createProduct</method>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Rice</value>
                                </argument>
                                <argument>
                                        <order>1</order>
                                        <value>1</value>
                                </argument>
                        </arguments>
                </variable>
        </pyfo>
    </pyfos>
</tui>
```

B.4 Petri

```
<?xml version="1.0" encoding="UTF-8"?>
<reseauPetri>
    <places>
        <place>
            <id>100</id>
            <name>
Task Cook Rice has to be performed. Take the pan to pour water in it.
            </name>
```

```
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>200</id>
                <name>
Pan is prepared to receive content. Pour water in the pan.
                </name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>300</id>
                <name>Send 10.0 water now in the pan.</name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>400</id>
                <name>
Water has been poured sucessfully. Put the pan on the heat.
                </name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>500</id>
                <name>Heat the water until it is boiling.</name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>600</id>
                <name>
Water is boiling. Pour rice into the pan with boiling water
 by puting the pan to reception zone.
                </name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>700</id>
                <name>Take the rice and put it to send zone.</name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>800</id>
```

```
                <name>Send 1.0 of rice.</name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>900</id>
                <name>Heat the pan until rice is cooked.</name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>1000</id>
                <name>Set the rice to cooked.</name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>1100</id>
                <name>The rice is cooked, pass it to the sieve.</name>
                <initialTokens>0</initialTokens>
        </place>
        <place>
                <id>1200</id>
                <name>The rice is prepared. Well done.</name>
                <initialTokens>0</initialTokens>
        </place>
</places>
<transitions>
        <transition>
                <id>100</id>
                <name>Task "Cook rice"</name>
        </transition>
        <transition>
                <id>200</id>
                <name>Pan is ready for receiving water</name>
        </transition>
        <transition>
                <id>300</id>
                <name>Water is being pouring in the pan</name>
        </transition>
        <transition>
                <id>400</id>
                <name>The pan has the right quantity of water</name>
        </transition>
```

```
            <transition>
                    <id>500</id>
                    <name>The pan is on the heat</name>
            </transition>
            <transition>
                    <id>600</id>
                    <name>Wait until the water is boiling</name>
            </transition>
            <transition>
                    <id>700</id>
                    <name>Put pan to receive rice</name>
            </transition>
            <transition>
                    <id>800</id>
                    <name>Pour rice to the pan</name>
            </transition>
            <transition>
                    <id>900</id>
                    <name>Continue heating the pan</name>
            </transition>
            <transition>
                    <id>1000</id>
                    <name>The rice is heating.</name>
            </transition>
            <transition>
                    <id>1100</id>
                    <name>The rice is cooked.</name>
            </transition>
            <transition>
                    <id>1200</id>
                    <name>The rice is prepared.</name>
            </transition>
    </transitions>
    <arcs>
            <arc>
                    <type>TransitionToPlace</type>
                    <from>100</from>
                    <to>100</to>
                    <n>1</n>
            </arc>
            <arc>
```

```
        <type>PlaceToTransition</type>
        <from>100</from>
        <to>200</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>200</from>
        <to>200</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>200</from>
        <to>300</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>300</from>
        <to>300</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>300</from>
        <to>400</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>400</from>
        <to>400</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>400</from>
        <to>500</to>
        <n>1</n>
</arc>
```

```
<arc>
        <type>TransitionToPlace</type>
        <from>500</from>
        <to>500</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>500</from>
        <to>600</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>600</from>
        <to>600</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>600</from>
        <to>700</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>700</from>
        <to>700</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>700</from>
        <to>800</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>800</from>
        <to>800</to>
        <n>1</n>
```

```
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>800</from>
        <to>900</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>900</from>
        <to>900</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>900</from>
        <to>1000</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>1000</from>
        <to>1000</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>1000</from>
        <to>1100</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>1100</from>
        <to>1100</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>1100</from>
        <to>1200</to>
```

```
                <n>1</n>
            </arc>
            <arc>
                <type>TransitionToPlace</type>
                <from>1200</from>
                <to>1200</to>
                <n>1</n>
            </arc>
        </arcs>
</reseauPetri>
```

B.5 Assistance

```
<?xml version="1.0" encoding="UTF-8"?>
<messages>
        <message>
                <message>startCookRice</message>
                <transition>100</transition>
        </message>
        <message>
                <message>
Name : Content of the pan ; Quantity : 0.0 ;
 Temperature : Cold, prepareToReceiveContent
                </message>
                <transition>200</transition>
        </message>
        <message>
                <message>
Name : Water ; Quantity : 1000.0 ;
 Temperature : Cold, prepareToSendContent
                </message>
                <transition>300</transition>
        </message>
        <message>
                <message>
Name : Water ; Quantity : 1000.0 ; Temperature : Cold,
 sendContent,Name : Content of the pan ; Quantity : 0.0 ;
 Temperature : Cold, 10.0
                </message>
                <transition>400</transition>
```

```
        </message>
        <message>
                <message>
Name : Content of the pan with Water ; Quantity : 10.0 ;
 Temperature : Cold, heat
                </message>
                <transition>500</transition>
        </message>
        <message>
                <message>
Name : Content of the pan with Water ; Quantity : 10.0 ;
 Temperature : Cold, setTemperature, Boiling
                </message>
                <transition>600</transition>
        </message>
        <message>
                <message>
Name : Content of the pan with Water ; Quantity : 10.0 ;
 Temperature : Boiling, prepareToReceiveContent
                </message>
                <transition>700</transition>
        </message>
        <message>
                <message>
Name : Rice ; Quantity : 1.0 ; Temperature : Cold,
 prepareToSendContent
                </message>
                <transition>800</transition>
        </message>
        <message>
                <message>
Name : Rice ; Quantity : 1.0 ; Temperature : Cold,
 sendContent, Name : Content of the pan with Water ;
 Quantity : 10.0 ; Temperature : Boiling, 1.0
                </message>
                <transition>900</transition>
        </message>
        <message>
                <message>
Name : Content of the pan with Water with Rice ;
 Quantity : 11.0 ; Temperature : Boiling, heat
```

```
                </message>
                <transition>1000</transition>
        </message>
        <message>
                <message>
Name : Content of the pan with Water with Rice ;
 Quantity : 11.0 ; Temperature : Boiling, setTemperature, Cooked
                </message>
                <transition>1100</transition>
        </message>
        <message>
                <message>
Name : Content of the pan with Water with Rice ;
 Quantity : 11.0 ; Temperature : Cooked, strain
                </message>
                <transition>1200</transition>
        </message>
</messages>
```

B.6 Affichages

```
<?xml version="1.0" encoding="UTF-8"?>
<outputs>
        <output>
                <name>Output1</name>
                <clazz>ca.usherbrooke.domus.output.console.ConsoleOutput</clazz>
                <arguments>
                </arguments>
        </output>
</outputs>
```

Annexe C

Fichiers de configuration pour l'application réelle

C.1 RFID

```
<?xml version="1.0" encoding="UTF-8"?>
<rfid>
        <tags>
                <tag>
                        <id>1234567890ABCDEF00000068</id>
                        <name>Main Pan</name>
                </tag>
                <tag>
                        <id>000000000000000000000005</id>
                        <name>Main Rice</name>
                </tag>
                <tag>
                        <id>1234567890ABCDEF00000069</id>
                        <name>Main Glass</name>
                </tag>
                <tag>
                        <id>1234567890ABCDEF0000006A</id>
                        <name>Main Strain</name>
                </tag>
        </tags>
        <readers>
                <reader>
                        <clazz>
```

```
ca.usherbrooke.domus.rfid.symbol.SymbolRfidTagReader
                        </clazz>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Main Washbasin</value>
                                </argument>
                                <argument>
                                        <order>1</order>
                                        <value>2</value>
                                </argument>
                        </arguments>
                </reader>
                <reader>
                        <clazz>
ca.usherbrooke.domus.rfid.symbol.SymbolRfidTagReader
                        </clazz>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Main Cupboard</value>
                                </argument>
                                <argument>
                                        <order>1</order>
                                        <value>1</value>
                                </argument>
                        </arguments>
                </reader>
                <reader>
                        <clazz>
ca.usherbrooke.domus.rfid.symbol.SymbolRfidTagReader
                        </clazz>
                        <arguments>
                                <argument>
                                        <order>0</order>
                                        <value>Main Stove</value>
                                </argument>
                                <argument>
                                        <order>1</order>
                                        <value>3</value>
                                </argument>
```

```
                        </arguments>
                </reader>
        </readers>
</rfid>
```

C.2 Placards

```
<?xml version="1.0" encoding="UTF-8"?>
<cupboards>
        <cupboard>
                <name>Placard riz</name>
                <code>0106</code>
                <open>P</open>
                <close>R</close>
        </cupboard>
        <cupboard>
                <name>Placard passoire</name>
                <code>0109</code>
                <open>P</open>
                <close>R</close>
        </cupboard>
        <cupboard>
                <name>Placard verre</name>
                <code>0113</code>
                <open>P</open>
                <close>R</close>
        </cupboard>
</cupboards>
```

C.3 Débitmétres

```
<?xml version="1.0" encoding="UTF-8"?>
<flowmeters>
        <flowmeter>
                <name>FlowMeterCuisineCold</name>
        </flowmeter>
        <flowmeter>
                <name>FlowMeterCuisineHot</name>
        </flowmeter>
```

```
</flowmeters>
```

C.4 TAC

```
<?xml version="1.0" encoding="UTF-8"?>
<tui xmlns:xdo="http://intranet.domus.usherbrooke.ca"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://intranet.domus.usherbrooke.ca
http://intranet.domus.usherbrooke.ca/fichiers/schemas/tui/modeltac.xsd">
        <tacs>
                <tac>
                        <name>1</name>
                        <token>Pan</token>
                        <constraints>
                                <constraint>Washbasin</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Pan
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>enterWashbasin</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>goingOutWashbasin</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>2</name>
                        <token>Pan</token>
                        <constraints>
                                <constraint>Stove</constraint>
                        </constraints>
```

```
                        <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Pan
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>enterStove</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>goingOutStove</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>3</name>
                        <token>Pan</token>
                        <constraints>
                                <constraint>Cupboard</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Pan
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>enterCupboard</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>goingOutCupboard</method>
                                </action>
                        </actions>
```

```
                </tac>
                <tac>
                        <name>4</name>
                        <token>Strain</token>
                        <constraints>
                                <constraint>Washbasin</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Strain
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>enterWashbasin</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>goingOutWashbasin</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>5</name>
                        <token>Pan</token>
                        <constraints>
                                <constraint>Strain</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Pan
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>strain</method>
                                </action>
```

```
                </actions>
        </tac>
        <tac>
                <name>6</name>
                <token>Strain</token>
                <constraints>
                        <constraint>Cupboard</constraint>
                </constraints>
                <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Strain
                </variable>
                <actions>
                        <action>
                                <name>add</name>
                                <automatic>true</automatic>
                                <listen />
                                <method>enterCupboard</method>
                        </action>
                        <action>
                                <name>remove</name>
                                <automatic>true</automatic>
                                <listen />
                                <method>goingOutCupboard</method>
                        </action>
                </actions>
        </tac>
        <tac>
                <name>7</name>
                <token>Glass</token>
                <constraints>
                        <constraint>Pan</constraint>
                </constraints>
                <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Glass
                </variable>
                <actions>
                        <action>
                                <name>add</name>
                                <automatic>true</automatic>
                                <listen />
                                <method>enterPan</method>
```

```
                        </action>
                    </actions>
                </tac>
                <tac>
                    <name>8</name>
                    <token>Rice</token>
                    <constraints>
                        <constraint>Glass</constraint>
                    </constraints>
                    <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Rice
                    </variable>
                    <actions>
                        <action>
                            <name>add</name>
                            <automatic>true</automatic>
                            <listen />
                            <method>enterGlass</method>
                        </action>
                    </actions>
                </tac>
                <tac>
                    <name>9</name>
                    <token>Rice</token>
                    <constraints>
                        <constraint>Cupboard</constraint>
                    </constraints>
                    <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Rice
                    </variable>
                    <actions>
                        <action>
                            <name>add</name>
                            <automatic>true</automatic>
                            <listen />
                            <method>enterCupboard</method>
                        </action>
                        <action>
                            <name>remove</name>
                            <automatic>true</automatic>
                            <listen />
```

```
                                        <method>goingOutCupboard</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>10</name>
                        <token>Glass</token>
                        <constraints>
                                <constraint>Cupboard</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Glass
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>enterCupboard</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>goingOutCupboard</method>
                                </action>
                        </actions>
                </tac>
                <tac>
                        <name>11</name>
                        <token>Glass</token>
                        <constraints>
                                <constraint>Washbasin</constraint>
                        </constraints>
                        <variable>
ca.usherbrooke.domus.kitchenaid.real.variable.Glass
                        </variable>
                        <actions>
                                <action>
                                        <name>add</name>
                                        <automatic>true</automatic>
```

```
                                        <listen />
                                        <method>enterWashbasin</method>
                                </action>
                                <action>
                                        <name>remove</name>
                                        <automatic>true</automatic>
                                        <listen />
                                        <method>goingOutWashbasin</method>
                                </action>
                        </actions>
                </tac>
        </tacs>
        <constraintsinstances>
                <constraintinstance>
                        <type>Cupboard</type>
                        <name>Main Cupboard</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.real.hardware.Cupboard
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Stove</type>
                        <name>Main Stove</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.real.hardware.Stove
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Washbasin</type>
                        <name>Main Washbasin</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.real.hardware.Washbasin
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Glass</type>
                        <name>Main Glass</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.real.hardware.Glass
                        </listen>
                </constraintinstance>
```

```
                <constraintinstance>
                        <type>Pan</type>
                        <name>Main Pan</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.real.hardware.Pan
                        </listen>
                </constraintinstance>
                <constraintinstance>
                        <type>Strain</type>
                        <name>Main Strain</name>
                        <listen>
ca.usherbrooke.domus.kitchenaid.real.hardware.Strain
                        </listen>
                </constraintinstance>
        </constraintsinstances>
        <pyfos>
                <pyfo>
                        <type>Pan</type>
                        <name>Main Pan</name>
                        <variable>
                                <factory>false</factory>
                                <clazz>
ca.usherbrooke.domus.kitchenaid.real.variable.Pan
                                </clazz>
                                <method />
                                <arguments />
                        </variable>
                </pyfo>
                <pyfo>
                        <type>Rice</type>
                        <name>Main Rice</name>
                        <variable>
                                <factory>false</factory>
                                <clazz>
ca.usherbrooke.domus.kitchenaid.real.variable.Rice
                                </clazz>
                                <method />
                                <arguments />
                        </variable>
                </pyfo>
                <pyfo>
```

```
                        <type>Strain</type>
                        <name>Main Strain</name>
                        <variable>
                                <factory>false</factory>
                                <clazz>
ca.usherbrooke.domus.kitchenaid.real.variable.Strain
                                </clazz>
                                <method />
                                <arguments />
                        </variable>
                </pyfo>
                <pyfo>
                        <type>Glass</type>
                        <name>Main Glass</name>
                        <variable>
                                <factory>false</factory>
                                <clazz>
ca.usherbrooke.domus.kitchenaid.real.variable.Glass
                                </clazz>
                                <method />
                                <arguments />
                        </variable>
                </pyfo>
        </pyfos>
</tui>
```

C.5 Réseau de Petri

```
<?xml version="1.0" encoding="UTF-8"?>
<reseauPetri>
        <places>
                <place>
                        <id>0</id>
                        <name>0</name>
                        <initialTokens>0</initialTokens>
                </place>
                <place>
                        <id>1</id>
                        <name>Main Pan is going out from Cupboard</name>
                        <initialTokens>0</initialTokens>
```

```
</place>
<place>
        <id>2</id>
        <name>Main Pan is entering in Washbasin</name>
        <initialTokens>0</initialTokens>
</place>
<place>
        <id>3</id>
        <name>Main Pan is going out from Washbasin</name>
        <initialTokens>0</initialTokens>
</place>
<place>
        <id>4</id>
        <name>Main Pan is entering in Kettle</name>
        <initialTokens>0</initialTokens>
</place>
<place>
        <id>5</id>
        <name>Main Glass is going out from Cupboard</name>
        <initialTokens>0</initialTokens>
</place>
<place>
        <id>6</id>
        <name>Main Rice is going out from Cupboard</name>
        <initialTokens>0</initialTokens>
</place>
<place>
        <id>7</id>
        <name>Main Rice is entering in Glass</name>
        <initialTokens>0</initialTokens>
</place>
<place>
        <id>8</id>
        <name>Main Rice is entering in Cupboard</name>
        <initialTokens>0</initialTokens>
</place>
<place>
        <id>9</id>
        <name>Main Glass is entering in Pan</name>
        <initialTokens>0</initialTokens>
</place>
```

```
            <place>
                    <id>10</id>
                    <name>Main Glass is entering in Washbasin</name>
                    <initialTokens>0</initialTokens>
            </place>
            <place>
                    <id>11</id>
                    <name>Main Glass is going out from Washbasin</name>
                    <initialTokens>0</initialTokens>
            </place>
            <place>
                    <id>12</id>
                    <name>Main Glass is entering in Cupboard</name>
                    <initialTokens>0</initialTokens>
            </place>
            <place>
                    <id>13</id>
                    <name>Main Strain is going out from Cupboard</name>
                    <initialTokens>0</initialTokens>
            </place>
            <place>
                    <id>14</id>
                    <name>Main Strain is entering in Washbasin</name>
                    <initialTokens>0</initialTokens>
            </place>
            <place>
                    <id>15</id>
                    <name>Main Pan is going out from Kettle</name>
                    <initialTokens>0</initialTokens>
            </place>
            <place>
                    <id>16</id>
                    <name>Main Pan is straining</name>
                    <initialTokens>0</initialTokens>
            </place>
    </places>
    <transitions>
            <transition>
                    <id>0</id>
                    <name>0</name>
            </transition>
```

```
<transition>
        <id>1</id>
        <name>1</name>
</transition>
<transition>
        <id>2</id>
        <name>2</name>
</transition>
<transition>
        <id>3</id>
        <name>3</name>
</transition>
<transition>
        <id>4</id>
        <name>4</name>
</transition>
<transition>
        <id>5</id>
        <name>5</name>
</transition>
<transition>
        <id>6</id>
        <name>6</name>
</transition>
<transition>
        <id>7</id>
        <name>7</name>
</transition>
<transition>
        <id>8</id>
        <name>8</name>
</transition>
<transition>
        <id>9</id>
        <name>9</name>
</transition>
<transition>
        <id>10</id>
        <name>10</name>
</transition>
<transition>
```

```
                <id>11</id>
                <name>11</name>
        </transition>
        <transition>
                <id>12</id>
                <name>12</name>
        </transition>
        <transition>
                <id>13</id>
                <name>13</name>
        </transition>
        <transition>
                <id>14</id>
                <name>14</name>
        </transition>
        <transition>
                <id>15</id>
                <name>15</name>
        </transition>
        <transition>
                <id>16</id>
                <name>16</name>
        </transition>
</transitions>
<arcs>
        <arc>
                <type>TransitionToPlace</type>
                <from>0</from>
                <to>0</to>
                <n>1</n>
        </arc>
        <arc>
                <type>PlaceToTransition</type>
                <from>0</from>
                <to>1</to>
                <n>1</n>
        </arc>
        <arc>
                <type>TransitionToPlace</type>
                <from>1</from>
                <to>1</to>
```

```
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>1</from>
        <to>2</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>2</from>
        <to>2</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>2</from>
        <to>3</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>3</from>
        <to>3</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>3</from>
        <to>4</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>4</from>
        <to>4</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>4</from>
```

```
        <to>5</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>5</from>
        <to>5</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>5</from>
        <to>6</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>6</from>
        <to>6</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>6</from>
        <to>7</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>7</from>
        <to>7</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>7</from>
        <to>8</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
```

```
        <from>8</from>
        <to>8</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>8</from>
        <to>9</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>9</from>
        <to>9</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>9</from>
        <to>10</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>10</from>
        <to>10</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>10</from>
        <to>11</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>11</from>
        <to>11</to>
        <n>1</n>
</arc>
<arc>
```

```
        <type>PlaceToTransition</type>
        <from>11</from>
        <to>12</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>12</from>
        <to>12</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>12</from>
        <to>13</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>13</from>
        <to>13</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>13</from>
        <to>14</to>
        <n>1</n>
</arc>
<arc>
        <type>TransitionToPlace</type>
        <from>14</from>
        <to>14</to>
        <n>1</n>
</arc>
<arc>
        <type>PlaceToTransition</type>
        <from>14</from>
        <to>15</to>
        <n>1</n>
</arc>
```

```
                <arc>
                        <type>TransitionToPlace</type>
                        <from>15</from>
                        <to>15</to>
                        <n>1</n>
                </arc>
                <arc>
                        <type>PlaceToTransition</type>
                        <from>15</from>
                        <to>16</to>
                        <n>1</n>
                </arc>
                <arc>
                        <type>TransitionToPlace</type>
                        <from>16</from>
                        <to>16</to>
                        <n>1</n>
                </arc>
        </arcs>
</reseauPetri>
```

C.6 Assistance

```
<?xml version="1.0" encoding="UTF-8"?>
<messages>
        <message>
                <message>okCookRice</message>
                <transition>0</transition>
        </message>
        <message>
                <message>Main Pan is going out from Cupboard</message>
                <transition>1</transition>
        </message>
        <message>
                <message>Main Pan is entering in Washbasin</message>
                <transition>2</transition>
        </message>
        <message>
                <message>Main Pan is going out from Washbasin</message>
                <transition>3</transition>
```

```
</message>
<message>
        <message>Main Pan is entering in Kettle</message>
        <transition>4</transition>
</message>
<message>
        <message>Main Glass is going out from Cupboard</message>
        <transition>5</transition>
</message>
<message>
        <message>Main Rice is going out from Cupboard</message>
        <transition>6</transition>
</message>
<message>
        <message>Main Rice is entering in Glass</message>
        <transition>7</transition>
</message>
<message>
        <message>Main Rice is entering in Cupboard</message>
        <transition>8</transition>
</message>
<message>
        <message>Main Glass is entering in Pan</message>
        <transition>9</transition>
</message>
<message>
        <message>Main Glass is entering in Washbasin</message>
        <transition>10</transition>
</message>
<message>
        <message>Main Glass is going out from Washbasin</message>
        <transition>11</transition>
</message>
<message>
        <message>Main Glass is entering in Cupboard</message>
        <transition>12</transition>
</message>
<message>
        <message>Main Strain is going out from Cupboard</message>
        <transition>13</transition>
</message>
```

```
        <message>
                <message>Main Strain is entering in Washbasin</message>
                <transition>14</transition>
        </message>
        <message>
                <message>Main Pan is going out from Kettle</message>
                <transition>15</transition>
        </message>
        <message>
                <message>Main Pan is straining</message>
                <transition>16</transition>
        </message>
</messages>
```

C.7 Affichages

```
<?xml version="1.0" encoding="UTF-8"?>
<outputs>
        <output>
                <name>Output1</name>
                <clazz>ca.usherbrooke.domus.output.proxy.OutputClient</clazz>
                <arguments>
                        <argument>
                                <order>0</order>
                                <value>localhost</value>
                        </argument>
                        <argument>
                                <order>1</order>
                                <value>1234</value>
                        </argument>
                </arguments>
        </output>
</outputs>
```

Annexe D

Article

Cet annexe contient la version publiée de mon article au workshop SmarTel 2007.

Annexe D : Article

Tangible User Interfaces for Cognitive Assistance

Baptiste Boussemart and Sylvain Giroux
Laboratoire DOMUS
University of Sherbrooke
2500, boul. de l'Universite, Sherbrooke, Quebec, Canada
{Baptiste.Boussemart, Sylvain.Giroux}@USherbrooke.ca

***Abstract*— Information technologies can help cognitively impaired people in planning and memory tasks. Though many projects already address such issues, they still use standard HCI and computers. Tangible User Interfaces (TUI) emerge as a new way to use a computer. They can improve the interactions and situatedness with respect to traditional HCI. In this paper, we present how we can use TUIs plan to build a cognitive assistant for activities of daily living on the form of an interactive table in a kitchen. Interactions are modelled with Token And Constraints paradigm (TAC), which may allow the user to have a better experience and to be more independent. We have implemented a Token and Constraint (TAC) layer using the capacities of object-oriented programming and XML architecture initialization.**

I. INTRODUCTION

Millions of people are suffering from cognitive deficits resulting from brain trauma or neurodegenerative diseases. In case of Traumatic Brain Injury (TBI), some parts of the brain may have been shocked due to a fall or a car accident. It causes loss of cognitive capacity, depending how damaged is the brain and the locations of these damages. In case of Alzheimer disease, tissues are deteriorated. Cognitive deficits increase over time as the disease progresses. Nonetheless in many cases they could stay at home if some assistance was provided.

Today many researches are based on the idea that a cognitive assistance could be provided with pervasive computers, but still relying on interfaces based on window, icon, menu, pointing (WIMP). But many studies show that such interfaces are not appropriate, because not enough simple for most cognitively impaired people [8]. Since the aimed cognitive assistant will have to support them in activities of daily living. Mark Weisers vision of the 21st century computer and ambient intelligence appears as the most suited approach [1]. Ambient Intelligence integrates ubiquitous computing ambient agents integrated into the environment; ubiquitous communication interactions between ambient agents; and intelligent and personalized HCI. Tangible User Interface (TUI) is a new approach to HCI that could provide the right answer to our problematic. TUI is defined as the association and manipulation of digital information to and through physical objects.

In this paper, cognitive assistance is first explained with typical problems and issues. Then requirements for HCI derived from cognitive issues are described. These requirements are then applied to TUIs. Finally a prototype of cognitive assistant is sketched, it is implemented as an interactive table. Interactions are exemplified through a scenario.

II. COGNITIVE ASSISTANCE

Cognition is related to brain mechanisms, like perception, recognition and reasoning. In order to assist cognitively impaired people, we need to understand some functions of the brain. To assist them, we intend to design and implement cognitive orthosis. A cognitive orthosis is an object designed specifically for rehabilitation purposes. In the present case, this object is a pervasive information system. It assists people in their activities of daily living (ADL). It has to be highly customizable to the needs of the individual. [2]

In this section a memory model is first explained. Then cognitive deficits are described. Finally some assistants are enumerated as solutions of these deficits.

A. Brain functions and memories

Cognition science has discovered brains functions and locations with studies on traumatic brain injured people. Many models have been inferred from these studies. We took information on these models from Wikipedia because they are summarized.

1) Prospective memory: Humans memorize planned actions or intentions in this memory. A cue, an event or a stimulus are the ways to recall events stored in this memory. Deficits in this memory will cause people not remembering what they have to do and what they have done. It is possible to enhance prospective memory with a to-do or grocery list. Because you may not remember what you have done, it causes repetitive questioning. [4] For example, an Alzheimer patient calling a relative every 10 minutes. Many artefacts may be helpful to address prospective memory deficits, for instance, an information system memorizing actions done by the user and warning him to not repeat them; iconic mnemonics to remember future intention by association; or a wearable medication dispenser.

2) Short-term memory: All information that will be processed by the working memory is "buffered" in the short-term memory, from 15 to 45 seconds. In order to remember a piece of information stored in short-term memory, humans have to use it many times and/or associate it with other concepts. Then, this piece of information tends to be stored in the long-term memory. Because it is short-term, interruptions will cause an automatic loss of this memory. When and how a system

should take users attention? In computer science, information redundancy [5] can be used use to make this information more accurate and easily memorizable.

3) Working memory: Information is processed in the working memory. Some studies show that humans work mainly phonologically and visually. For example, when you read you "speak in your brain" or you imagine visually the scene or both. Pictures, videos and sounds are processed more easily, but cannot replace completely texts. Illustrated contents are a way to put easily understandable contents, such as pictures and contents that are more precise, such as texts. Although Internet satisfies well to these criteria, hyper-texts tend to produce cognitive overload. [8]

4) Long-term memory: Long-term information, like memories or knowledge, is stored in the long-term memory as a structure. A disease or a shock may cause problem in searching for information in this memory, as a consequence of a broken structure.

5) Procedural memory: Procedural memory is a part of long-term memory. It stores procedures like recipes. It defines your skills. Identifying missing information may help people to remember the steps they need to do. A non-intrusive procedural assistant that process information in parallel, to the user and that can be checked or questioned by the user, could be a solution. Such a system has then to remember how to perform a step, what actions user has already completed, etc.

B. Cognitive deficits

Memory impairments is due to physical damages that came from accidents, like falls and car accidents, or diseases such as Alzheimers disease. These impairments decrease functions of the brain at different degrees: attention, planning, language. Such impairments may cause anxiety and stress in people with cognitive deficits and decrease motivation.

1) Attentional deficits: Attention deficits may be caused by short-term memory weaknesses. For assistance, lights and sounds can attract users attention depending on the context [7]. In some cases, e.g. fire risks, user distractions must not be tolerated and the assistance system has to redirect immediately user attention and even undertake actions on user behalf, e.g. calling emergency services.

2) Planning deficits: Prospective memory stores planned actions and cues are needed to remember plans. An agenda may be a solution to assist people in planning tasks [4]. But too often they lack dynamicity and then are not suitable for assisting Activities of Daily Living (ADLs). Multi-tasking is also barely supported because it is linked with the attention skills. It demands planning, timing, monitoring and control of action skills. All these skills may decrease as well. [4]

3) Language abilities: Delays in finding words increase with aging [4]. Moreover a limited amount of information can be processed in parallel with people above 65 years old. Aged people tend to use more simple sentences and trying to understand hard sentences can cause anxiety [8]. Reading on a screen takes more time than reading on paper. Understanding shows also the same issues [8].

C. Consequences of cognitive deficits on personality

Anxiety and stress may be caused by cognitive deficits and by inappropriate assistance [8, 4, 7]. Anti-social behaviours could then emerge. Assisting becomes very complex because it depends on a person history, culture and tastes. As a result cognitively impaired people may reject assistive technologies we intend to build. Pat of the solution may lie in improving HCI with respect to the user cognitive impairments.

Motivation is also important for quality of life. It is related on how well needs are satisfied. Five categories of desire have been identified: physiological, safety, social, esteem and actualization (http://en.wikipedia.org/wiki/Maslow's_hierarchy). People with cognitive deficits can easily lack of security and esteem, for instance, when a person often forgets to turn off the oven or if he doubts in its capacity to accomplish a task, because he knows that he failed many times before.

D. Cognitive assistant

Intelligent information system based on context awareness, artificial intelligence and procedural assistance is a promising direction to help cognitively impaired people [3, 4, 7, 9]. Assistance is then built upon actions like anticipations, interaction and recuperation [5]. Because designers of such systems must have ethical cares, some guidelines must be respected [5]:

1) Do not intend to replace users actions, but to assist them.
2) Assistance must be personalized, exploiting the user profile, to make the assistance accurate, not stressful and acceptable.
3) Assistant must not be intrusive.
4) Users may have some behavioural troubles. Deal with them, do not to stress users.
5) Do not propose something that the person can not do.

III. HCI REQUIREMENTS FOR COGNITIVE ASSISTANCE

Designing HCI is not an easy task. Participatory [10], human-centred and system-centred design have all their own answer to how the design of HCI should proceed. System-centred design does not involve users, opposite to participatory design. Ambient-intelligence usually adopts a human-centred design approach. A ubiquitous system has to integrate seamlessly into user tasks. Ideally the design of an intelligent assistive home should be tailored specifically for this resident. Personalization then becomes a central issue. Special attention has to be paid to language too because the resident and the pervasive cognitive assistant need to communicate with each other. Stress and anxiety of users may be avoided thanks to a correct designed HCI that personalize interactions.

User-centred design is a good way to build an efficient HCI for people with cognitive deficits [9, 12, 3]. User profiles are used to configure the system. A user profile exhibits a unique combination of strength and weaknesses [2]. It describes the users abilities [4], the users preferences, tastes, and the health profile (stage of the disease, traumatic brain injuries, etc.).

Interactions must be straightforward, transparent [3] and intuitive. The more possible inputs and outputs the system has, the more complex the HCI will be [5]. Adaptive interfaces

change with respect to the program state, like context. They do not seem to be appropriate for cognitively impaired people [9]. But they may be required for specific functions as procedural interfaces and user personalization. If the user understands why the interface has changed, then the use of adaptive interfaces would be OK. In the best case, interfaces should not change without the user consent.

IV. Procedural Assistance

HCIs for procedural interfaces are different than common HCIs found in desktop systems. They straightforwardly represent tasks that people need to do. The interface displays the tasks and the current step. When the user has completed a step, the interface shows up the next step, helping gradually the user to perform a task [5]. Procedural interfaces are very useful for people with cognitive deficits because this kind of assistant reminds all the time what to do, how to do it and when it is done. Coupled with an agenda, it can be a very effective tool for cognitive assistance. Moreover the users concentration and motivation increase as well. A procedural assistant can reveal very helpful to a user in doing his ADLs. In our work, we build some models of procedures, like kitchen recipes. [9]

V. Meaning, Language and Notification

It is important that the system and the user understand each other very well. HCI for computer uses as metaphor the desktop, folder, trash, paper. But an intelligent home is not a desktop computer. What would be the best metaphors, the best language? [4] There is no definitive answer.

Language used by the system must be simple, be it text, iconic language [4], speech or handwriting. Examples of principles are: confirming a noun with visual representation, confirming a verb with a video of the action, presenting a set of synonyms, etc. [4] Sets of colours are a good mean to express alerts or various system states [5, 8, 20].

When appropriate, the system has to catch users attention. People tend to dislike systems that interrupt them too often and may reject the technology [15, 7]. Ambient notification is used in many projects and commercial products. Smooth sound and light transitions may not disturb people too much (http://www.manovich.net/IA/). TUIs use objects to replace interactions through mouse and keyboard.

VI. Tangible User Interface as a Solution

The standard combination of mouse, screen and keyboard is too complex for user with cognitive deficits [16]. Tangible User Interfaces (TUIs) is a new HCI approach that uses physical objects, not virtual ones. Virtual information is associated with these objects and manipulations of these objects will manipulate information. For instance, a key would become able to unlock a door and a computer, as its purpose is to give access to allowed people. TUIs are more intuitive and offer a better metaphor than common HCI, as they use real objects instead of virtual widgets.

Many sensors can be integrated in the environment, like infrared location sensors, electromagnetic contacts on doors, pressure rugs TUIs enable to improve ubiquitous information systems by embedding sensors into objects. Indeed tagging objects is an easy technical issue for making TUIs. It is possible to identify what is the user doing, infer where are objects and augment reality with RFID enabled personal digital assistant (PDA).

A. TUI's models

There are three main categories of TUIs: interactive table, constructive assembly, and token and constraint (TAC) [17]. Interactive tables allow user to interact with a system by moving objects on a table. Vision and video projections are used, e.g. www.sonicforms.org. Constructive assembly is inspired from lego-like block to drive interactions. The assembly is often used to organize information, create trees or build robots, for instance web.media.mit.edu/ hayes/topobo/. TAC defines a clear constructive model for HCI [19]. TAC is influenced by the MCRpd model [21], which departs from the standard Model-View-Controller (MVC) model in that the controller and the view are physical representation. Yet the HCIs have to model how information is linked to physical objects, with easy understandable metaphors.

TAC is the most serious proposal we have seen so far to provide for straightforward, explicit, and simple interactions, a compulsory requirement for pervasive cognitive assistance. Interactive tables are not pervasive as they use video cameras and projection display for a table specifically. HCIs are therefore very specific and a unified set of interactions is hard to define. Constructive assembly does not responding to any of the requirements described previously.

B. Token and Constraints paradigm

In TAC, a token is a physical object, which has an associated variables. Tokens can be put on constraints. A constraint is a physical element, which limits the physical movements of the token. The token associated with a constraint is called a TAC, defining a set of possible interactions. For example the token can physically translate on the constraint. The translation will invoke actions on the associated variable to the token. The TAC is enabled when the token and the constraint are associated, enabling indeed the interactions defined in the TAC.

VII. TUIs and Cognitive Assistance

TUIs principles can be applied in many ways to provide for cognitive assistance. In this section, we sketch three of them, namely everyday objects, iconic cards, and interactive paper.

A. Building on top of everyday objects

Every day objects can be enhanced with sensors for cognitive assistance. For instance, Sentient Artefacts [22] project has transformed mirror, tooth brush, etc. into intelligent objects.

Kitchen is the place where people live. They prepare food, have social interaction [20]. Preparing food is a difficult task for people with cognitive deficits. So kitchen is a good place where people can be motivated and foster their self-esteem when they succeed. A procedural assistant for kitchen would

surely be of great value. The procedural assistant could monitor the person in food preparation relying on RFID technology to identify objects and their locations and to implement TACs features.

B. Iconic cards as a mean of interaction

According to TUI philosophy, a user should express his requests directly to the system without using a keyboard and a screen for feedback. Oralys (www.oralys.ca) and others companies are providing tools that enable a new type of communication with symbols. The idea is to replace mouse and keyboard with iconic cards. An interaction base will then read cards with printed symbols. Bases stand for actions (verbs) and cards for objects (nouns). We used the TAC model for defining interactions. When the user put a card on the base, a continuous or discrete action is triggered on a virtual variable. Immediate feedback is also provided. If the action is continuous then the action will stop when the card is taken off. "Get information", "help", "locate", "call" are some examples of bases that that could help people.

C. Interactive paper

Interactive paper is directly inspired by MCRpd model, and we intend to extend it to TAC. The paper would contain a RFID chip, which allows coupling the paper to virtual information. We propose to use this type of paper in a manner similar to iconic cards. For instance, a RFID printer will propose a task to the user, for instance a recipe. The user sees what the task is about with information printed out, but he wants more information. He can then put the task on the "Get information" base in order to obtain complementary details on the task described on the paper. If the user needs to know where the task has to be performed, he will use the "Locate" base. To know how to perform the task, he can invoke the interactive cognitive assistant by using an "Help" base.

D. Clear unambiguous consistent metaphors

Tangible papers, cards and objects can be used as alternative means to control and interact with the assistant. Naturally they tend to have the same meaning, but there may be differences sometimes. When putting a coffee card on the "locate" base, the assistant will help to find the coffee pot. But when the base is associated to the coffee pot itself, it should help to locate the coffee machine. So defining the possibilities of the environment and the relations between objects are important.

E. Modular tangible interfaces

To provide for feedback, we intend to use phidgets. Phidgets are USB artifacts that can be combined. Using Phidgets, we will propose patterns of objects, alike in sentient artifacts. A customizable clock built with Phidgets servo-motor is one example of such a pattern. Each clock can have many meanings and will be put at strategic locations in the home. These clocks will replaces screens and provide feedback for specific TACs. We expect this approach to ease the deployment of a intelligent home system.

TAC	Constraints	Token	Variable	Actions	Feedback
1	Help	Task	Task	Add	Display help for achieving the task
				Remove	Hide the help
				Up	Scroll Up
				Down	Scroll Down
2	Get information	Task	Task	Add	Display a description of the task
				Remove	Hide the description
				Up	Scroll Up
				Down	Scroll Down
3	Drop	Task	Task	Add	Delete the task
4	Get information	Container	Content	Add	Display a description of the content
				Remove	Hide the description
				Up	Scroll Up
				Down	Scroll Down
5	Pour, Destination	Container	Content	Add	Prepare to receive content; display signal when ready
6	Pour, Source	Container	Content	Add	When destination container is ready; display quantity to pour
				Remove	Pour the quantity to the destination
				Up	Increase quantity
				Down	Decrease quantity
7	Help	Container	Content	Add	Display an help in context of the task
				Remove	Hide the help
				Up	Scroll Up
				Down	Scroll Down
8	Drop	Container	Content	Add	Empty the container
9	Achat	Container	Content	Add	Buy ingredient
10	Mix	Container	Content	Add	Mix content
11	Heat	Container	Content	Add	Display the temperature to set
				Remove	Set the temperature to the content
				Up	Increase temperature of the content
				Down	Decrease temperature of the content

Fig. 1. TAC model for the prototype

Phidgets proposes USB artifacts that can be assembled. Therefore many combinations are possible. But proposing some patterns of objects, like sentient artifacts, is a way to facilitate deployment of a intelligent home system.

A customizable clock with Phidgets servo is one example of pattern. Each clock can have many meanings and placed everywhere in home. It replaces screens, when information is every time the same.

VIII. PROJECT

In previous section, many opportunities to develop TUIs for cognitive assistance at home were presented. This section will discuss a procedural assistant currently under development for kitchens and based on TUI. The development of the prototype proceeds in two steps. The first step consists in building an interactive table using TAC interactions (Fig.1). The second step is to integrate the same interactions as with the table into the kitchen.

A. A TAC model for a kitchen assistant

We defined a TAC model for preparing meal (Fig.2). Basic actions were modeled: pouring, mixing, cutting, heating...

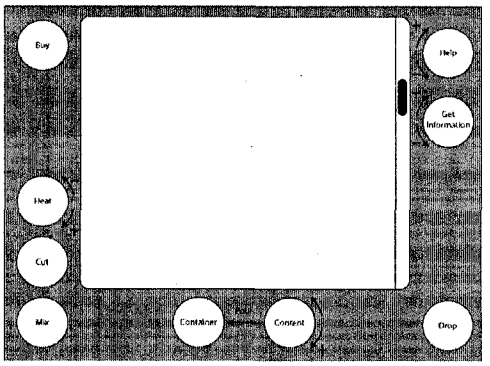

Fig. 2. Screen of the prototype

Then TAC assistive actions were added to provide for help: help and get information. Finally buy and drop actions were added whenever containers are full or not.

Content type is a preparation or an ingredient. For example a preparation cannot be bought. As content is an associated variable to a container, a physical block, some actions will do nothing, but a signal will tell us that nothing have been done. Some cases are not explicit in the table, therefore some feedbacks need to be designed as well: a red light that tell us this action is not possible.

B. Scenario

Let use a scenario to get a glimpse on how the prototype works. The user has to prepare cooked rice. An object represents this task and the variable is set to the first step. First the user wants to know what is planned: he puts this object on the "Get Information" base.

When he knows what to do, he takes a pan and puts water in. In the prototype, he takes a block representing an empty container and puts it on the "pour source (Container)" base. He then takes a block representing water and puts it on the "pour destination (Content)". He turns clockwise the block and set the quantity of water to 1 liter. He takes the water, and the quantity is virtually poured in the container block. The user can verify the contents of a block with the "Get information" base. The base will say: "1 liter of cold water".

In a real environment, RFID tags would identify objects; sensors on the tap will be used to infer what kind of activity is done. The activity of pouring contents cannot be modeled as a TUI, but in the prototype we simulate the same action. Detecting how much water or another fluid have been poured is yet a problem. Gyroscopic sensor may be a mean to detect the angle and time, so the system could infer the poured quantity.

Next the user puts the pan on heating plate and waits until the water boils. In the prototype, the user puts the container on the "Heat" base and sets the temperature by turning the block. In a real environment, temperature sensors integrated in the oven can be used. The state of the rice is displayed.

Some rice is then put in the boiling water, using the same steps as when water was put in the pan.

The user then waits until the rice is cooked. The container must be on the "heat" zone. A fast timer is used in the prototype. "Get information" on the container will tell that it contains boiling water with rice in and the state of the rice: not cooked, well cooked and too much cooked. An interval of time is allowed for the user to take away the container from the heat base.

If the user does not know how to perform a step of the task, he can use the help base. The help feature will direct his attention to specific objects needed to perform the task. The help feature will also show where the object should be. Since we use an interactive table, it is very easy: arrows will point from objects to their destination.

C. Technical details and current status of the prototype

Vision is used for detecting user's actions. A TAC layer is built on a sensor layer, which uses ReacTiVision (http://mtg.upf.edu/reactable) vision software from the ReacTable project.

The prototype is not finished yet as the procedural assistant is not yet implemented. But actions on virtual variable are finished.

A video projector and a video camera are needed for completing the table, but a simulator of the ReacTiVision exists. It allows running the prototype without a true interactive table.

The TAC model is complete. We have implemented a layer, which is initialized with an XML file specifying the TAC table of Figure 1. The interface is also described in XML and linked to the TAC layer.

D. Integration with the real world

Many technical issues are still to be solved. RFID has a great potential to address most of these problems. Now we are working on how we will transfer the prototype to the real assistant. The related issues have been discussed in the scenario section.

IX. Related projects

Recipetable (http://www.recipetable.net) is a project that proposes recipes taking into account the ingredients on the interactive table. The HCI is also tactile and it is used for selecting recipes. Its purpose is to help people in deciding which recipes are available depending available ingredients.

Cognitive Cubes [23] is based on the MediaBlocks. The system shows a cognitive cube assembly and the user has to build the same assembly with MediaBlocks. Each cube has blue tape in some edges, therefore defines only one assembly. The solution rotates slowly in a screen, so the user can examine each face of the solution that he should obtain. MediaBlocks allows the system to measure user's accuracy and rapidity, therefore cognition abilities of the user. It is not an assistance, but it demonstrates how TUI can be used for cognition purposes.

X. Conclusion

Human Computer Interaction has been pointed as an important step of design of a cognitive assistance. In this paper, we tried to figure out what are the requirements of a well designed HCI, and why todays desktop HCIs are not appropriate for cognitive assistance. For this, memory and brain impaired were re-examined. Then we proposed Tangible User Interface as a solution for an intelligent home HCI. The TAC customizable framework was selected to build a prototype TUI for cognitive assistance in the kitchen. The first was to design and simulate an interactive table. Interactions on this table simulate actions of preparing a recipe. We plan to integrate this prototype in a pervasive kitchen, using RFID tags and other sensors.

References

[1] M. Weiser, *The computer of 21st century* 1991.

[2] E. F. LoPresti and A. Mihailidis and N. Kirsch, *Assistive technology for cognitive rehabilitation: State of the art* Psychology Press Ltd, 2004.

[3] M. Rudary and S. Singh and M. E. Pollack, *Adaptive Cognitive Orthotics: Combining Reinforcement Learning and Constraint-Based Temporal Reasoning* Proceedings of the 21st International Conference on Marchine Learning, 2004.

[4] K. Hawkey and K. M. Inkpen and K. Rockwood and M. McAllister and J. Slonim, *Requirements Gathering with Alzheimer's Patients and Caregivers* ASSETS'05, ACM, 2005.

[5] G. A. Boy *Procedural Interfaces* IHM, ACM, 2002.

[6] N. Alm and A. Astell and M. Ellis and R. Dye and G. Gowans and J. Campbell, *A cognitive prosthesis and communication support for people with dementia* 2004.

[7] A. Oulasvirta and A. Salovaara, *A Cognitive Meta-Analysis of Design Approches to Interruptions in Intelligent Environments* CHI, ACM, 2004.

[8] S. Singh, *Designing Intelligent Interfaces for Users with Memory and Language Limitations* Aphasiology, vol. 14, 2000.

[9] J. A. Jorge, *Adaptive Tools for the Elderly, New Devices to cope with Age-Induced Cognitive Disabilities* WUAUC'01, ACM, 2001.

[10] M. Wu and R. Baecker and B. Richards, *Participatory Design of an Orientation Aid for Amnesics* CHI, ACM, 2005.

[11] P. Gorman and R. Dayle and C. Hood and L. Rumrell, *Effectiveness of the ISAAC cognitive prosthetic system for improving rehabilitation outcomes with neurofunctional impairment* IIOS Press, 2003.

[12] J. Paradise and E. D. Mynatt and C. Williams and J. Goldthwaite, *Designing a Cognitive Aid for the Home: A Case-Study Approach* ASSETS '04, ACM, 2004.

[13] E. LoPresti and R. Simpson and N. Kirsch and D. Schreckenghost, *Solo: Interactive Task Guidance* ASSETS'05, ACM, 2005.

[14] A. Voinikonis and K. Irmscher and H. Schulze, *Distributed processing of reminding tasks within the mobile memory aid system, MEMOS* UMICS, 2004.

[15] T. Matthews and A. K. Dey and J. Mankoff and S. Carter and T. Rattenbury, *A Toolkit for Managing User Attention in Peripheral Displays* UIST '04, ACM, 2004.

[16] A. Mihailidis and G. R. Fernie, *Context-aware assistive devices for older adults with dementia* Gerontechnology, 2002; 2(2): 173-189.

[17] B. Ullmer and H. Ishii and R. J. K. Jacob, *Token+Constraint Systems for Tangible Interaction with Digital Information* ACM, 2005.

[18] E. H. Calvillo-Gmez and N. Leland and O. Shaer and R. J. K. Jacob, *The TAC Paradigm: Unified Conceptual Framework to Represent Tangible User Interfaces* ACM.

[19] A. Svensk, *Design for Cognitive Assistance* Licentiate Thesis CERTEC, Lth Number 1:2001, 2001.

[20] J. C.-H. Lee, *Spatial User Interfaces: Augmenting Human Sensibilities in a Domestic Kitchen* MIT thesis, 2005.

[21] B. Ullmer and H. Ishii, *Emerging Frameworks for Tangible User Interfaces* IBM System Journal, Vol. 39, No. 3/4, 2000.

[22] F. Kawsar and K. Fujinami and T. Nakajima, *Experiences with Developing Context-Aware Applications with Augmenting Artefacts* ubiPCMM, 2005.

[23] E. Sharlin, Y. Itoh, B. Watson, Y. Kitamura, S. Sutphen, L. Liuo, *Cognitive Cubes: A Tangible User Interface for Cognitive Assessment* Hands-On Interfaces, CHI, 2002.

Bibliographie

[1] Alissa N. Antle. The cti framework : informing the design of tangible systems for children. *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 195–202, New York, NY, USA, 2007. ACM.

[2] Nicolas Baskiotis and Nicolas Ferey. Introduction to ambient intelligence (ami). Web. http ://www.limsi.fr/intelligence.htm.

[3] Jérémy Bauchet. Modèle pour la reconnaissance d'activités. Master's thesis, Universite de Sherbrooke, 2006.

[4] Jérémy Bauchet and André Mayers. A modelisation of adls in its environment for cognitive assistance. *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, pages 139–146, New York, NY, USA, 2005. ACM.

[5] Guy A. Boy. Cognitive function analysis for human-centered automation of safety-critical systems. *CHI '98 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[6] Guy A. Boy. Interfaces procédurales. *IHM '02 : Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine)*, pages 81–88, New York, NY, USA, 2002. ACM.

[7] Andreas Butz, Michael Schmitz, Antonio Krüger, and Harald Hullmann. Tangible uis for media control : probes into the design space. *CHI '05 : CHI '05 extended abstracts on Human factors in computing systems*, pages 957–971, New York, NY, USA, 2005. ACM.

[8] Eduardo H. Calvillo-Gámez, Nancy Leland, Orit Shaer, and Robert J. K. Jacob. The tac paradigm : unified conceptual framework to represent tangible user interfaces. *CLIHC '03 : Proceedings of the Latin American conference on Human-computer interaction*, pages 9–15, New York, NY, USA, 2003. ACM.

[9] Andrew Carvey, Jim Gouldstone, Pallavi Vedurumudi, Adam Whiton, and Hiroshi Ishii. Rubber shark as user interface. *CHI '06 : CHI '06 extended abstracts on Human factors in computing systems*, pages 634–639, New York, NY, USA, 2006. ACM.

[10] Lee Chia-Hsun, Ma Yu-Pin, and Jeng Taysheng. A spatially-aware tangible interface for computer-aided design. *CHI '03 : CHI '03 extended abstracts on Human factors in computing systems*, pages 960–961, New York, NY, USA, 2003. ACM.

[11] E. Dishman. Inventing wellness systems for aging in place. *Computer*, 37(5) :34–41, May 2004.

[12] Institut du vieillissement. Le système de santé survivra-t-il au vieillissement de la population ? http ://www.cihr-irsc.gc.ca/f/10518.html.

[13] Richard Etter and Carsten Röcker. A tangible user interface for multi-user awareness systems. *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 11–12, New York, NY, USA, 2007. ACM.

[14] Kenneth P. Fishkin. A taxonomy for and analysis of tangible interfaces. *Personal Ubiquitous Comput.*, 8(5) :347–358, 2004.

[15] Audrey Girouard, Erin Treacy Solovey, Leanne M. Hirshfield, Stacey Ecott, Orit Shaer, and Robert J. K. Jacob. Smart blocks : a tangible mathematical manipulative. *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 183–186, New York, NY, USA, 2007. ACM.

[16] Saul Greenberg and Chester Fitchett. Phidgets : easy development of physical interfaces through physical widgets. *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 209–218, New York, NY, USA, 2001. ACM.

[17] Kirstie Hawkey, Kori M. Inkpen, Kenneth Rockwood, Michael McAllister, and Jacob Slonim. Requirements gathering with alzheimer's patients and caregivers. *Assets '05 : Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pages 142–149, New York, NY, USA, 2005. ACM.

[18] James A. Hendler and Paul Roller Michaelis. The effects of limited grammar on interactive natural language. *CHI '83 : Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 190–192, New York, NY, USA, 1983. ACM.

[19] Eva Hornecker. A design theme for tangible interaction : embodied facilitation. *ECSCW'05 : Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work*, pages 23–43, New York, NY, USA, 2005. Springer-Verlag New York, Inc.

[20] Eva Hornecker. Physicality in tangible interaction : Bodies and the world. *First International Workshop on Physicality*, Lancaster University, 6-7 Feb 2006.

[21] Eva Hornecker and Jacob Buur. Getting a grip on tangible interaction : a framework on physical space and social interaction. *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 437–446, New York, NY, USA, 2006. ACM.

[22] Chen-Je Huang. Not just intuitive : examining the basic manipulation of tangible user interfaces. *CHI '04 : CHI '04 extended abstracts on Human factors in computing systems*, pages 1387–1390, New York, NY, USA, 2004. ACM.

[23] Jörn Hurtienne and Johann Habakuk Israel. Image schemas and their metaphorical extensions : intuitive patterns for tangible interaction. *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 127–134, New York, NY, USA, 2007. ACM.

[24] H. Ishii, C. Ratti, B. Piper, Y. Wang, A. Biderman, and E. Ben-Joseph. Bringing clay and sand into digital design — continuous tangible user interfaces. *BT Technology Journal*, 22(4) :287–299, 2004.

[25] Hiroshi Ishii and Brygg Ullmer. Tangible bits : towards seamless interfaces between people, bits and atoms. *CHI '97 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM.

[26] Robert J. K. Jacob, Hiroshi Ishii, Gian Pangaro, and James Patten. A tangible interface for organizing information using a grid. *CHI '02 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 339–346, New York, NY, USA, 2002. ACM.

[27] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reactable : exploring the synergy between live music performance and tabletop tangible interfaces. *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 139–146, New York, NY, USA, 2007. ACM.

[28] Joaquim A Jorge. Adaptive tools for the elderly : new devices to cope with age-induced cognitive disabilities. *WUAUC'01 : Proceedings of the 2001 EC/NSF workshop on Universal accessibility of ubiquitous computing*, pages 66–70, New York, NY, USA, 2001. ACM.

[29] Martin Kaltenbrunner and Ross Bencina. reactivision : a computer-vision framework for table-based tangible interaction. *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 69–74, New York, NY, USA, 2007. ACM.

[30] Neung Eun Kang and Wan Chul Yoon. A cognitive modeling of the user's exploratory behavior with prior knowledge. *TAMODIA '05 : Proceedings of the 4th international workshop on Task models and diagrams*, pages 35–42, New York, NY, USA, 2005. ACM.

[31] Fahim Kawsar, Kaori Fujinami, and Tatsuo Nakajima. Augmenting everyday life with sentient artefacts. *sOc-EUSAI '05 : Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 141–146, New York, NY, USA, 2005. ACM.

[32] Scott R. Klemmer, Mark W. Newman, Ryan Farrell, Mark Bilezikjian, and James A. Landay. The designers' outpost : A tangible interface for collaborative web site design, 2001.

[33] Matthias Kranz, Dominik Schmidt, Paul Holleis, and Albrecht Schmidt. A display cube as a tangible user interface, 2005.

[34] Sami Laakso. Tangible user interfaces and interruptions, in acm repository, 2001.

[35] Jackie Chia-Hsun Lee. *Spatial User Interfaces : Augmenting Human Sensibilities in a Domestic Kitchen.* PhD thesis, Massachusetts Institute of Technology, 2005.

[36] Surapong Lertsithichai and Matthew Seegmiller. Cubik : a bi-directional tangible modeling interface. *CHI '02 : CHI '02 extended abstracts on Human factors in computing systems*, pages 756–757, New York, NY, USA, 2002. ACM.

[37] Edmund LoPresti, Ned Kirsch, Richard Simpson, and Debra Schreckenghost. Solo : interactive task guidance. *Assets '05 : Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pages 190–191, New York, NY, USA, 2005. ACM.

[38] George F. Luger. *Artificial Intelligence; Structure and Strategies for Complex Problem Solving.* Addison Wesley, 2002.

[39] Dennis Maciuszek, Johan Aberg, and Nahid Shahmehri. What help do older people need? : constructing a functional design space of electronic assistive technology applications. *Assets '05 : Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pages 4–11, New York, NY, USA, 2005. ACM.

[40] Paul Marshall. Do tangible interfaces enhance learning? *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 163–170, New York, NY, USA, 2007. ACM.

[41] Tara Matthews, Anind K. Dey, Jennifer Mankoff, Scott Carter, and Tye Rattenbury. A toolkit for managing user attention in peripheral displays. *UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 247–256, New York, NY, USA, 2004. ACM.

[42] Stuart Mealing and Masoud Yazdani from Department of Computer Science University of Exeter England. A computer-based iconic language. http ://www.intellectbooks.com/iconic/com_base/icon-2.htm, 1990.

[43] Alex Mihailidis and Geoffrey R. Fernie. Context-aware assistive devices for older adults with dementia. *Gerontechnology*, 2(2) :173–189, 2002.

[44] Kerhuel N. Vieillissement et habitat ; recherche comparée sur les politiques de l'habitat en direction des personnes vieillissantes et en perte d'autonomie. Technical report, Ministère de l'Equipement, des Transports et du Logement, 2001.

[45] Antti Oulasvirta and Antti Salovaara. A cognitive meta-analysis of design approaches to interruptions in intelligent environments. *CHI '04 : CHI '04 extended abstracts on Human factors in computing systems*, pages 1155–1158, New York, NY, USA, 2004. ACM.

[46] Sharon Oviatt. Human-centered design meets cognitive load theory : designing interfaces that help people think. *MULTIMEDIA '06 : Proceedings of the 14th annual ACM international conference on Multimedia*, pages 871–880, New York, NY, USA, 2006. ACM.

[47] Blandine Paccoud, David Pache, Hélène Pigot, and Sylvain Giroux. Report on the impact of a user-centered approach and usability studies for designing mobile and context-aware cognitive orthosis. *Pervasive Computing for Quality of Life Enhancement*, 4541/2007 :179–187, 2007.

[48] Gian Pangaro, Dan Maynes-Aminzade, and Hiroshi Ishii. The actuated workbench : computer-controlled actuation in tabletop tangible interfaces. *UIST '02 : Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 181–190, New York, NY, USA, 2002. ACM.

[49] Jessica Paradise, Elizabeth D. Mynatt, Cliff Williams, and John Goldthwaite. Designing a cognitive aid for the home : a case-study approach. *Assets '04 : Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*, pages 140–146, New York, NY, USA, 2004. ACM.

[50] Dominique Paret. Technical state of art of "radio frequency identification – rfid" and implications regarding standardization, regulations, human exposure, privacy. *sOcEUSAI '05 : Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 9–11, New York, NY, USA, 2005. ACM.

[51] James Patten, Hiroshi Ishii, Jim Hines, and Gian Pangaro. Sensetable : a wireless object tracking platform for tangible user interfaces. *CHI '01 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 253–260, New York, NY, USA, 2001. ACM.

[52] James Patten, Ben Recht, and Hiroshi Ishii. Audiopad : a tag-based interface for musical performance. *NIME '02 : Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–6, Singapore, Singapore, 2002. National University of Singapore.

[53] Elin Ronby Pedersen, Tomas Sokoler, and Les Nelson. Paperbuttons : expanding a tangible user interface. *DIS '00 : Proceedings of the 3rd conference on Designing interactive systems*, pages 216–223, New York, NY, USA, 2000. ACM.

[54] Wen Qi, Jean-Bernard Martens, Robert van Liere, and Arjan Kok. Reach the virtual environment : 3d tangible interaction with scientific data. *OZCHI '05 : Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*, pages 1–10, Narrabundah, Australia, Australia, 2005. Computer-Human Interaction Special Interest Group (CHISIG) of Australia.

[55] Hayes Raffle, Hiroshi Ishii, and James Tichenor. Super cilia skin : A textural interface. *Textile*, 2. 3 :1–19, 2004.

[56] Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. Topobo : a constructive assembly system with kinetic memory. *CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 647–654, New York, NY, USA, 2004. ACM.

[57] Jun Rekimoto, Brygg Ullmer, and Haruo Oba. Datatiles : a modular platform for mixed physical and graphical interactions. *CHI '01 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 269–276, New York, NY, USA, 2001. ACM.

[58] Adele Rochon, Helene Pigot, and Beaulieu. Etude des besoins technologiques favorisant l'autonomie des personnes ayant des déficits cognitifs. Technical report, Universite de Sherbrooke, laboratoire DOMUS, 2005.

[59] Dan Rosenfeld, Michael Zawadzki, Jeremi Sudol, and Ken Perlin. Physical objects as bidirectional user interface elements. *Emerging Technologies*, 0272-1716 :44–49, 2004.

[60] Matthew Rudary, Satinder Singh, and Martha E. Pollack. Adaptive cognitive orthotics : Combining reinforcement learning and constraint-based temporal reasoning. *21st International Conference on Machine Learning*, 2004.

[61] Christian Sandor and Gudrun Klinker. A rapid prototyping software infrastructure for user interfaces in ubiquitous augmented reality. *Pers Ubiquit Comput*, 9 :169–185, 2005.

[62] Orit Shaer, Nancy Leland, Eduardo H. Calvillo-Gamez, and Robert J. K. Jacob. The tac paradigm : specifying tangible user interfaces. *Personal Ubiquitous Comput.*, 8(5) :359–369, 2004.

[63] Ehud Sharlin, Yuichi Itoh, Benjamin Watson, Yoshifumi Kitamura, Steve Sutphen, and Lili Liu. Cognitive cubes : a tangible user interface for cognitive assessment. *CHI '02 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 347–354, New York, NY, USA, 2002. ACM.

[64] Sameer Singh. Designing intelligent interfaces for users with memory and language limitations. *Aphasiology*, 14 :157–177, 2000.

[65] Joshua R. Smith, Kenneth P. Fishkin, Bing Jiang, Alexander Mamishev, Matthai Philipose, Adam D. Rea, Sumit Roy, and Kishore Sundara-Rajan. Rfid-based techniques for human-activity detection. *Commun. ACM*, 48(9) :39–44, 2005.

[66] R. W. Smith. Providing natural language assistance in locating objects : a general model for information selection and generation. *IEA/AIE '88 : Proceedings of the 1st international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 922–931, New York, NY, USA, 1988. ACM.

[67] Brygg Ullmer and Hiroshi Ishii. Emerging frameworks for tangible user interfaces. *IBM Systems Journal*, 39 :3–4, 2001.

[68] Brygg Ullmer, Hiroshi Ishii, and Robert J. K. Jacob. Token+constraint systems for tangible interaction with digital information. *ACM Trans. Comput.-Hum. Interact.*, 12(1) :81–118, 2005.

[69] John Underkoffler and Hiroshi Ishii. Illuminating light : an optical design tool with a luminous-tangible interface. *CHI '98 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 542–549, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[70] John Underkoffler, Brygg Ullmer, and Hiroshi Ishii. Emancipated pixels : real-world graphics in the luminous room. *SIGGRAPH '99 : Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 385–392, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[71] Anne C. van Rossum. *Visual Iconic Language : A Framework for Visual Iconic Languages.* PhD thesis, Delft University of Technology, 2006.

[72] Denis Vergnès, Sylvain Giroux, and Daniel Chamberland-Tremblay. Interactive assistant for activities of daily living. IOS Press, editor, *From Smart Home to Smart Care*, volume 15, pages 229–236, 2005.

[73] Manuela Waldner, Jörg Hauber, Jürgen Zauner, Michael Haller, and Mark Billinghurst. Tangible tiles : design and evaluation of a tangible user interface in a collaborative tabletop setup. *OZCHI '06 : Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction : design : activities, artefacts and environments*, pages 151–158, New York, NY, USA, 2006. ACM.

[74] Ryoichi Watanabe, Yuichi Itoh, Masatsugu Asai, Yoshifumi Kitamura, Fumio Kishino, and Hideo Kikuchi. The soul of activecube : implementing a flexible, multimodal, three-dimensional spatial tangible interface. *Comput. Entertain.*, 2(4) :15–15, 2004.

[75] Mark Weiser. *The computer for the 21st century*, pages 933–940. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[76] Mike Wu, Ron Baecker, and Brian Richards. Participatory design of an orientation aid for amnesics. *CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 511–520, New York, NY, USA, 2005. ACM.

[77] Jamie Zigelbaum, Michael S. Horn, Orit Shaer, and Robert J. K. Jacob. The tangible video editor : collaborative video editing with active tokens. *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 43–46, New York, NY, USA, 2007. ACM.

[78] Babak Ziraknejad. *TIH - Tangible Interfaces at Home An exploration of computationally enhanced interactive devices.* PhD thesis, University of Washington, 2004.