Description des tâches avec un système interactif multiutilisateur et multimodal : Etude comparative de notations

Frédéric Jourde, Yann Laurillau, Laurence Nigay

Description des tâches avec un système interactif multiutilisateur et multimodal : Etude comparative de notations

FREDERIC JOURDE
Université Grenoble-Alpes, LIG, CNRS, Grenoble
YANN LAURILLAU
Université Grenoble-Alpes, LIG, CNRS, Grenoble
LAURENCE NIGAY
Université Grenoble-Alpes, LIG, CNRS, Grenoble

Résumé : De nombreux systèmes interactifs, professionnels ou grand public, permettent conjointement l'interaction multiutilisateur et multimodale. Un système interactif est multimodal lorsqu'un utilisateur peut interagir avec le système par l'usage de plusieurs modalités d'interaction (en entrée ou en sortie) de façon parallèle ou non. Nous constatons que de plus en plus de systèmes multiutilisateurs ou collecticiels sont multimodaux, comme ceux construits autour d'une surface interactive et les consoles de jeu de type Wii ou Xbox. Nous traitons dans cet article de la description des tâches-utilisateur avec de tels systèmes interactifs multiutilisateurs et multimodaux. Précisément, nous dressons un panorama des notations existantes permettant la description des tâches mono ou multi-utilisateur avec une attention particulière pour les notations à base d'arbre de tâches. Nous focalisons aussi sur les tâches élémentaires ou actions mono/multi-modales de l'utilisateur en considérant les notations de description de l'interaction multimodale. Pour cela, nous proposons une étude comparative d'un ensemble de notations de description selon une grille d'analyse regroupant des concepts généraux à l'interaction et des concepts propres à l'interaction multiutilisateur et multimodale.

Mots clés : tâche,, multiutilisateur, collecticiel, multimodalité, modèle de tâche, notation de description.

Abstract: Multi-user multimodal interactive systems involve multiple users who can use multiple interaction modalities. Multi-user multimodal systems are becoming more prevalent, especially systems based on large shared multi-touch surfaces or video game centers such as Wii or Xbox. In this article we address the description of the tasks with such interactive systems. We review existing notations for the description of tasks with a multi-user multimodal interactive system and focus particularly on tree-based notations. For elementary tasks (e.g. actions), we also consider the notations that describe multimodal interaction. The contribution is then a comparison of existing notations based on a set of organized concepts. While some concepts are general to any notation, other concepts are specific to human-computer interaction, or to multi-user interaction and finally to multimodal interaction.

Keywords:, task, multi-user, groupware, multimodality, task model, description notation.

Adresse des auteurs :
Frédéric Jourde (fjourde.seigneurs@gmail.com), Université Grenoble-Alpes, LIG, FR-38000, Grenoble, France.
Yann Laurillau ({yann.laurillau}@imag.fr), Université Grenoble-Alpes, LIG, FR-38000, Grenoble, France.
Laurence Nigay ({laurence.nigay}@imag.fr), Université Grenoble-Alpes, LIG, FR-38000, Grenoble, France.

1. INTRODUCTION

De nombreux systèmes interactifs, grand public ou professionnels, permettent conjointement l'interaction multiutilisateur et multimodale comme ceux décrits dans [Meija 2007 ; Safin 2009 ; Tse 2008]. Un système interactif est multimodal lorsqu'un utilisateur peut interagir avec le système par l'usage de plusieurs modalités d'interaction (en entrée ou en sortie) en parallèle ou non. Aussi, nous observons que de plus en plus de systèmes interactifs multiutilisateurs (également nommés collecticiels) sont multimodaux, comme ceux construits autour d'une surface interactive [Tse 2008] (Figure 3) et les consoles de jeu de type Wii ou Xbox. Par exemple, Tse et coll. [Tse 2008] ont développé une version multiutilisateur et multimodale du célèbre jeu Warcraft : équipés d'un casque-micro, deux joueurs coopèrent en face à face pour contrôler des unités de guerriers en interagissant avec une table interactive multitouche : les deux modalités en entrée pour réaliser les actions du jeu sont l'interaction gestuelle sur la table et des commandes vocales. Au-delà des prototypes issus de la recherche, un nombre croissant de systèmes interactifs multiutilisateurs et multimodaux sont développés dans de nombreux domaines d'application comme le médical [Meija 2007], ou la conception industrielle [Safin 2009]. Ce dernier constat souligne le besoin de notations de description de systèmes interactifs multiutilisateurs et multimodaux.

Parmi les outils à disposition du concepteur du système interactif, nous nous focalisons sur **les notations de description de la tâche**. Nous verrons que plusieurs buts sont identifiés pour une notation à destination du concepteur que ce soit pour la conception, la génération ou l'évaluation d'un système interactif [Balbo 2004].

Aussi dans cet article, nous dressons un panorama des notations existantes permettant la description des tâches de l'utilisateur avec un système interactif multiutilisateur et multimodal, avec une attention particulière pour les notations à base d'**arbre de tâches** très nombreuses dans le domaine de l'Interaction Homme-Machine (IHM). Il convient de noter que pour les tâches élémentaires ou actions mono/multi-modales de l'utilisateur nous considérons aussi les notations de description de l'interaction multimodale autorisant ces actions multimodales. Pour cela, nous proposons une étude comparative d'un ensemble de notations existantes, permettant implicitement ou explicitement de décrire la tâche, à partir d'une liste de critères, mettant en évidence similarités et différences **conceptuelles**. Comme dans [Limbourg 2004], l'objectif est de comparer les notations selon les concepts explicitement véhiculés par la notation. Cette comparaison est utile dans le domaine de l'IHM pour dresser un panorama des notations et mettre en évidence celles qui véhiculent les concepts spécifiques aux tâches interactives multiutilisateur et multimodales. Notre étude comparative permet donc aussi d'identifier les notations qui nécessiteraient d'être étendues pour couvrir les aspects liés à l'interaction multiutilisateur et multimodale. Contrairement à d'autres taxonomies de notations [Balbo 2004 ; Brun 1995], notre étude comparative ne permet pas d'évaluer les notations mais souligne les variations en termes de concepts explicitement présents dans les notations. En effet pour une évaluation des notations dans le but de servir de guide aux concepteurs pour le choix d'une notation, d'autres critères sont à prendre en compte comme la lisibilité, la facilité d'apprentissage et l'extensibilité [Balbo 2004 ; Brun 1995].

L'article est organisé de la façon suivante. La partie 2 établit une grille d'analyse regroupant des critères afin de comparer les notations selon quatre facettes : caractéristiques générales, interaction utilisateur, interaction multiutilisateur et interaction multimodale. La partie 3 présente les principaux points caractérisant des

notations existantes, issues aussi bien de la psychologie cognitive que du domaine de l'Interaction Homme-Machine (IHM), du Travail Collaboratif Assisté par Ordinateur (TCAO), de l'interaction MultiModale (MM) ou du Génie Logiciel (GL). Avant de conclure, la partie 4, sur la base de deux tableaux, présente des éléments de synthèse comparative de notations existantes puis une illustration et analyse comparative des deux notations Dynamo-Aid et COMM qui explicitent les concepts liés à l'interaction multiutilisateur et multimodale.

2. ETUDE COMPARATIVE : CRITÈRES

Afin de cartographier le paysage des notations dédiées à la description des systèmes interactifs multiutilisateurs et multimodaux, nous avons identifié et organisé un ensemble de critères pour établir une classification, en nous appuyant sur des critères empruntés en grande partie à [Limbourg 2004 ; Jourde 2008 ; Molina 2009]. Bien que relatifs au pouvoir expressif d'une notation [Brun 1995 ; Balbo 2004] mais sans aborder les autres aspects d'évaluation d'une notation, les critères retenus visent à mettre en évidence les variations conceptuelles ou syntaxiques [Limbourg 2003] entre notations. Pour cela, nous identifions quatre classes de critères, du plus général au plus spécifique (multiutilisateur et multimodalité) :

- *Définition de la notation* : les contours de la notation et de son applicabilité ;
- *Description de l'interaction utilisateur* : aspects de la notation permettant l'expression l'interaction utilisateur ;
- *Description de l'interaction multiutilisateur* : aspects spécifiques de la notation pour exprimer l'interaction multiutilisateur ;
- *Description de l'interaction multimodale* : aspects de la notation permettant l'expression de l'interaction multimodale.

2.1 Définition de la notation

Cinq critères sont identifiés pour caractériser de façon générale une notation : champ disciplinaire, couverture du processus de conception, formes et types de représentations du langage d'expression de la notation, outils instrumentant la notation.

Champ disciplinaire. Ce critère précise le domaine dont la notation est issue.

Plusieurs domaines de recherche (psychologie cognitive, ethnographie, ingénierie logicielle, etc.) proposent des notations au champ d'application varié. Par exemple, la notation K-MAD [Lucquiaud 2005], issue de la psychologie cognitive, vise à analyser la tâche de l'utilisateur dans son activité en lien avec son environnement. La notation CTT [Paternó 1999], issue de l'ingénierie logicielle, est dédiée à la description des tâches de l'utilisateur avec le système interactif : de tels arbres de tâches peuvent être exploités conjointement avec d'autres modèles comme le modèle du domaine applicatif pour générer une interface homme-machine (par exemple, à partir des modèles UsiXML).

Couverture. Dans [Brun 1995 ; Balbo 2004] les notations sont positionnées selon les grandes étapes d'un cycle de développement logiciel (analyse des besoins, analyse de la tâche, spécification, conception générale, conception détaillée, support à la génération de code ou prototype), comme le cycle en V [Rook 1986]. Une notation peut avoir plusieurs usages tout au long du cycle de développement d'un système interactif. Cet usage peut être très en amont lors de l'analyse des besoins ou en aval pour conduire une

évaluation heuristique des descriptions. Par exemple, la notation CTT [Paternó 1999] couvre à la fois l'analyse de la tâche mais aussi l'implémentation du système interactif par transformation de modèles. La notation GTA [Van der Veer 2000] s'inscrit quant à elle dans le cadre de la méthode DUTCH qui s'étend depuis l'étude ethnographique jusqu'à la description de l'interaction concrète avec la notation NUAN [Venema 1999] (extension de la notation UAN [Hartson 1990]). Enfin certaines notations comme ICOM [Dragicevic 2004], ICARE [Bouchet 2004] et ICO [Navarre 2009] permettent l'exécution de l'interaction multimodale. Basé sur ICARE, ACICARE [Serrano 2006] est un environnement pour le développement mais aussi pour l'évaluation expérimentale utilisateur.

Au delà des étapes du processus de développement, nous considérons comme critères les éléments d'un système interactif qui sont décrits par la notation, en nous référant au modèle ARCH [Bass 1992] présenté à la Figure 1 : noyau fonctionnel, adaptateur de noyau fonctionnel, contrôleur de dialogue, interaction abstraite et interaction concrète. Nous considérons si une notation permet de décrire la partie fonctionnelle NF (noyau fonctionnel, adaptateur de noyau fonctionnel) d'un système interactif, la partie gestion de l'interaction CD (contrôleur de dialogue), la partie interaction abstraite IA et la partie interaction concrète IC.

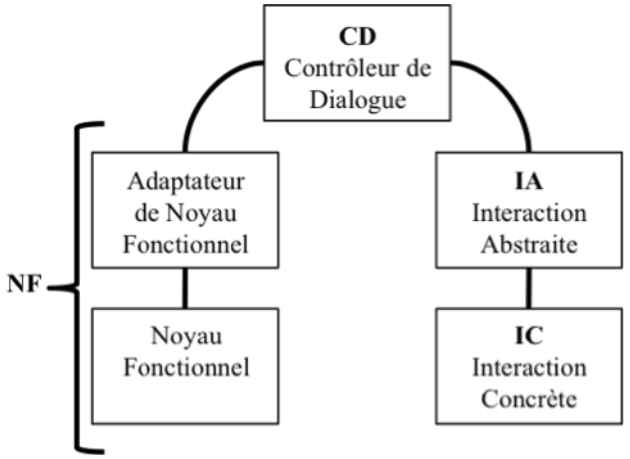


Fig. 1. Modèle ARCH [Bass 1992] pour distinguer quatre éléments d'un système interactif que décrit une notation : NF, CD, IA et IC.

Représentations. Ce critère est relatif au langage visuel de la notation. Sont pris en compte, la variété de vues (type et nombre) pour exprimer différents aspects de la description ainsi que les formes de représentation (textuelle, tabulaire, graphique, diagramme dont les structures arborescentes).

Par exemple, la notation GTA [Van der Veer 2000] offre plusieurs vues fournissant des vues statiques (par exemple, un arbre de tâches) et des vues dynamiques (par exemple, diagramme d'activité) du système interactif. Les formes de représentation de ces vues sont variées et incluent des arbres ou une structure tabulaire pour la description de l'interaction concrète concernant les tâches élémentaires ou actions de l'utilisateur (avec la notation NUAN [Venema 1999]).

Modèle de la notation. Ce critère indique la nature du modèle qui explicite la notation. Ce modèle peut être formel/semi-formel (réseau de Petri, méta-modèle, schéma XML, etc.) ou informel (indications textuelles).

Instrumentation. Ce critère indique si la notation est outillée. La complexité d'une notation rend souvent indispensable de disposer d'un outil logiciel permettant d'éditer la description. L'outil d'édition renforce l'usage et une meilleure compréhension de la notation, mais aussi facilite les échanges entre les utilisateurs de la notation tout en contribuant à sa popularisation. Enfin, les descriptions éditées selon une notation peuvent éventuellement constituer une entrée pour réaliser la génération de code comme préalable à la phase de d'implémentation.

Les critères relatifs au champ disciplinaire, au modèle et à l'instrumentation font écho aux critères d'origine et de formalisation de [Limbourg 2003] et d'outil de modélisation de [Molina 2009].

2.2 Description des tâches

Trois critères sont identifiés pour caractériser la description des tâches en général : structure de la représentation des tâches utilisateur, composition des tâches, représentations pour exprimer les objets du domaine.

Structuration. Ce critère précise la façon dont la notation articule les tâches utilisateur et système : structure hiérarchique, flot d'activité, arbre de tâches, séquence, machine à état, etc.

Une structure largement répandue est un arbre de tâches décrivant un ensemble de tâches composées de sous-tâches. Les feuilles d'un tel arbre ou tâches élémentaires (par exemple, les actions) précisent l'interaction concrète et peuvent faire l'objet d'une description dédiée additionnelle, comme l'usage de la notation NUAN [Venema 1999] dans le cadre de la notation GTA [Van der Veer 2000].

Composition. Ce critère précise les mécanismes employés par la notation pour exprimer l'agencement logique ou temporel des tâches utilisateurs au sein de la structure qui les expose. Cet agencement peut être exprimé par un plan d'exécution, des opérateurs logiques ou temporels, ou encore des conditions.

Les opérateurs pour exprimer la composition qui sont les plus souvent employés incluent : l'exécution en séquence ou en parallèle, l'alternative et l'exclusivité. Les notations offrent une gamme plus ou moins large d'opérateurs. Par exemple, les opérateurs de la notation CTT [Paternó 1999] sont issus du langage formel de description LOTOS [ISO 1989]. Au contraire la notation GTA [Van der Veer 2000] laisse le concepteur libre de définir ses propres opérateurs.

Objets du domaine. Ce critère précise comment une notation permet la description des objets du domaine. Comme la description de l'interaction, la notation peut s'appuyer sur une notation complémentaire pour décrire le modèle du domaine, par exemple un diagramme de classe UML, ou proposer sa propre représentation. Ce critère précise également si la description des objets du domaine fait l'objet d'une représentation indépendante ou intégrée.

Par exemple, la notation GTA [Van der Veer 2000] propose sa propre représentation, comme vue indépendante, pour décrire les objets du domaine tandis que

la notation CTML [Wurdel 2008] repose explicitement sur le diagramme de classe UML. La notation CIAN [Molina 2007] associe la description des objets du domaine à chaque tâche au sein d'une représentation unique.

Ces critères sont relatifs, respectivement, aux critères de niveau d'opérationnalisation, de planification des tâches et d'objets manipulés de [Limbourg 2004].

2.3 Description des tâches multiutilisateur

Cette section concerne les concepts pour comparer les notations dans leur capacité conceptuelle pour décrire les tâches multiutilisateurs. Pour cela, nous nous appuyons sur l'ontologie de van Welie et coll. [van Welie 1998]. Cette ontologie identifie cinq concepts fondamentaux : rôle, acteur, tâche, objet et événement.

Les concepts de rôle, d'acteur ou agent, de groupe, de tâche coopérative/collaborative, d'objet partagé et de droits d'accès sont spécifiques à la description d'un système interactif multiutilisateur :

- *Rôle* : le concept de rôle permet l'association de classes d'acteurs en charge d'une tâche à réaliser, sans se préoccuper de savoir exactement qui, c'est-à-dire un acteur, sera en charge de la réalisation.
- *Acteur* : un humain ou un agent système exécutant une tâche pour un rôle donné.
- *Tâche* : identifie une série d'activités à accomplir pour atteindre un but dans un rôle donné.

  En particulier, nous précisons dans quelle mesure une notation permet la distinction entre les deux grandes classes d'activité de groupe : coopération et collaboration. Comparée à une activité coopérative, une activité collaborative offre un caractère plus opportuniste du fait d'une absence de division planifiée du travail. Roschelle et Teasley [Roschelle 1995] distinguent la coopération de la collaboration comme suit : "*Cooperative work is accomplished by the division of labour among participants, as an activity where each person is responsible for a portion of the problem solving. [...] Collaboration [is a] mutual engagement of participants in a coordinated effort solve the problem together*".

  De nombreuses notations comme la notation CCTT [Paternó 1998] permettent la description de tâches multiutilisateurs coopératives. La notation CIAN [Molina 2007] est une des rares notations distinguant coopération et collaboration par l'introduction de types de tâche multiutilisateur idoines.
- *Objet* : identifie une information ou un artefact (objet du domaine) manipulé pour réaliser une tâche donnée. La notion d'objet partagé (artefact dans la terminologie de A. Dix [Dix 2004]) est importante pour étudier des propriétés comme la conscience de groupe (*group awareness*).
- *Evènement* : un événement modélise la dynamique d'une tâche et qui influence son déroulement.

Ces concepts sont également pris en compte par le critère d'aspect de la collaboration dans [Limbourg 2004] et par la modélisation du travail de groupe de [Molina 2009].

2.4 Description des tâches multimodales

La multimodalité implique des actions de l'utilisateur ou du système selon plusieurs modalités d'interaction. La multimodalité concerne les tâches élémentaires, feuilles de l'arbre de tâches. Pour étudier la description de ces tâches élémentaires multimodales, nous considérons les notations de description de l'interaction multimodale autorisant ces actions multimodales. Ainsi cette section concerne les concepts pour comparer les notations dans leur capacité conceptuelle pour décrire l'interaction multimodale. Pour cela, nous nous appuyons sur la définition de [Nigay 1995] pour caractériser une modalité selon un couple <dispositif d'interaction, langage d'interaction>. Quatre concepts fondamentaux sont alors identifiés pour préciser l'interaction multimodale : modalité, dispositif d'interaction, langage d'interaction et composition de modalités.

Modalité. Ce concept identifie si une notation permet l'expression d'une modalité ou combinaison de modalités.

Basée sur CTT [Paternó 1999], la notation Dynamo-AID [Clerckx 2007] est une notation exprimant cette notion de modalité en l'associant à une tâche élémentaire (feuille d'un arbre de tâches) pour préciser l'interaction concrète.

Dispositif. Ce concept identifie si une notation intègre la notion de dispositif et permet d'associer son usage à la réalisation d'une tâche élémentaire.

De nombreuses notations intègrent cette notion avec une granularité plus ou moins fine, comme simplement nommer un dispositif ou bien décrire son comportement à l'aide de diagrammes état-transition.

Langage. Ce concept identifie la capacité des notations à décrire la manière d'utiliser un dispositif d'interaction, c'est-à-dire à décrire des primitives d'utilisation (actions élémentaires) ou des tâches concrètes correspondant à l'usage des dispositifs.

Composition. Ce concept identifie les notations permettant l'expression de compositions de modalités pour décrire l'interaction multimodale. Ces compositions peuvent être précisées par les propriétés CARE et des opérateurs spécifiant des contraintes temporelles de la composition multimodale [Serrano 2009].

Les propriétés CARE (Complémentarité, Assignation, Redondance, Equivalence) [Coutaz 1995] permettent de décrire la composition de modalités utilisées pour réaliser une tâche élémentaire concrète. La complémentarité désigne l'usage conjoint de deux ou plusieurs modalités nécessaires à la transmission des informations pour réaliser une tâche. Une autre composition CARE, la redondance, signifie que plusieurs modalités sont nécessaires pour accomplir une tâche et que chacune d'elles porte l'intégralité de l'information. La complémentarité et la redondance de modalités impliquent au niveau système la fusion d'informations [Lalanne 2009] provenant de plusieurs modalités. Par exemple la notation NUAN [Venema 1999] permet de décrire la combinaison d'événements (et donc la fusion à réaliser par le système) pour des entrées clavier/souris. Mais la description des interactions est limitée aux modalités en entrée dont les dispositifs sont le clavier et la souris.

3. NOTATIONS EXISTANTES : ÉTAT DE L'ART

Ayant identifié les critères et concepts pour guider l'étude des notations de description existantes, cette partie présente les notations existantes étudiées. Ces notations relèvent de plusieurs domaines. La Figure 2 illustre la cartographie des notations étudiées dans cet article du point de vue de leur champ disciplinaire. Certaines héritent ou emploient des notations issues du Génie Logiciel comme UML (Figure 2). Une large part **des notations étudiées relèvent de l'IHM** ou sont communes à d'autres domaines comme la Psychologie Cognitive. De plus, une grande part de ces notations sont à base d'arbre de tâches, révélatrice des pratiques et de leur ancrage fort dans le domaine de l'IHM. Les notations UAN [Hartson 1990], NUAN [Venema 1999] et HTA [Annett 1967] sont mentionnées dans le schéma de la Figure 2 pour mettre en évidence les relations d'héritage mais ne font pas partie des notations étudiées dans cet état de l'art. En particulier, la notation HTA [Annett 1967] est mentionnée car identifiée comme une des toutes premières notations à base d'arbre de tâches, et ayant inspiré les notations existantes.

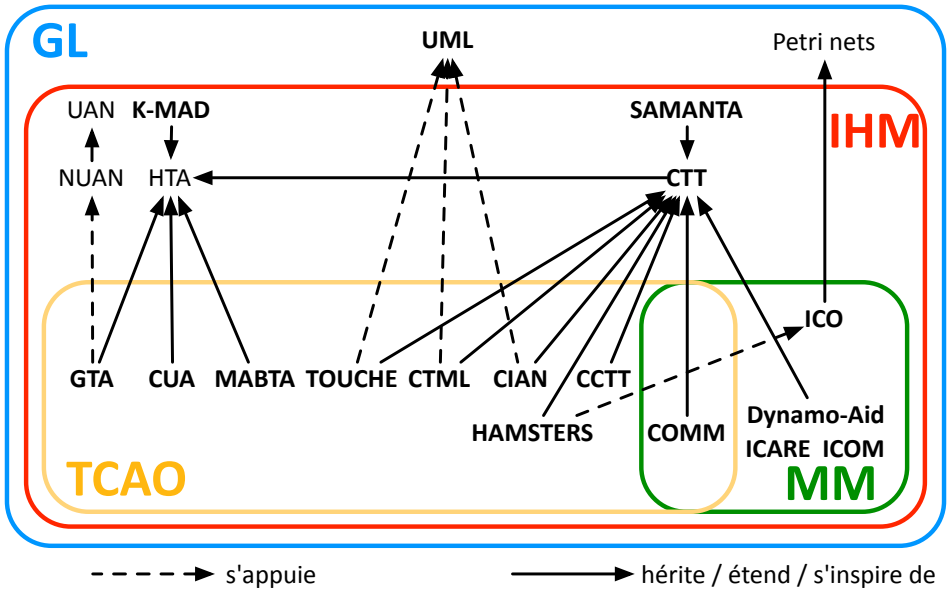


Fig. 2. Relations entre les notations étudiées, en fonction de leur champ disciplinaire. GL : Génie Logiciel. IHM : Interaction Homme-Machine. TCAO : Travail collaboratif Assisté par Ordinateur. MM : MultiModalité.

Dans le cadre de cette étude sur la description des tâches interactives multiutilisateurs et multimodales, d'autres notations sont décrites dans [Jourde 2011]. L'étude englobant plusieurs axes du domaine de l'IHM, certaines notations ne sont pas décrites : par exemple c'est le cas des notations IOG [Carr 1994] ou Nimmit [Vanacken 2007] qui ont une approche à base de machine à états pour, respectivement, la description de l'interaction utilisateur et la description de l'interaction multimodale. De nombreux principes de IOG et Nimmit se recoupent avec la notation ICO [Navarre 2009] (paragraphe 3.3.3) dont le formalisme sous-jascent, les réseaux de Petri, englobe ces approches à base de machine à états.

Dans la suite de cette partie sur les notations existantes, la première section aborde les notations généralistes pour la description des tâches interactives. La seconde section est consacrée aux notations dédiées aux tâches multiutilisateur et la troisième section aux notations dédiées aux tâches élémentaires concrètes (actions utilisateur) multimodales.

3.1 Notations pour la description de tâches

3.1.1 CTT

CTT (*Concurrent Task Tree*) est une notation d'arbre de tâches destinée à l'analyse des besoins et à la description de tâches mono-utilisateur [Paternò 1999]. CTT couvre les composants de contrôleur de dialogue et d'interaction abstraite du modèle ARCH [Bass 1992]. L'éditeur CTTE [Mori 2002] est un environnement interactif pour la description d'un système interactif avec CTT.

La notation CTT introduit quatre types de tâches : les tâches utilisateur, les tâches abstraites pouvant être décomposées en sous-tâches, les tâches système, les tâches mentales. Les relations entre les tâches sont définies par des opérateurs hérités de LOTOS [ISO 1989] comme la concurrence (`|||`), l'activation (>>), l'alternative (`[]`) ou l'opérateur unaire pour l'itération (`*`). Aussi, CTT bénéficie d'une description formelle de ces opérateurs. De plus, la particularité de CTT est d'exprimer la composition entre tâches descendantes d'une tâche parent plutôt que de le préciser sur le plan de la tâche parent.

3.1.2 SAMANTA

La méthodologie SAMANTA (*Situation Awareness Modeling and ANalysis for Transition Amelioration*) [Villaren 2012a ; Villaren 2012b] promeut une approche pour concevoir des interfaces tenant mieux compte de l'impact cognitif sur l'utilisateur de transitions entre tâches jugées complexes. Aussi, cette méthodologie repose sur l'usage conjoint de modèles couplant modèle de tâches et de description de contexte sous la forme de réseaux d'éléments de situation. Les tâches sont décrites avec la notation CTT (voir section 3.1.1). Le modèle d'éléments de situation est un graphe cartographiant des connaissances liées aux situations de réalisation des tâches et obtenus par une analyse amont, par exemple au travers de RETEX (retours d'expérience).

Du point de vue de la description des tâches, les concepts sont donc ceux de la notation CTT.

3.1.3 K-MAD

K-MAD (*Kernel of Model for Activity Description*) [Lucquiaud 2005] est une notation héritant de nombreuses notations existantes (Diane+ [Tarby 2001], MAD [Scapin 1990], CTT [Paternò 1999], et GTA [Van der Veer 2000]) car sa construction repose sur une analyse approfondie de celles-ci par extraction de leurs caractéristiques intrinsèques. Du point de vue du modèle ARCH [Bass 1992], K-MAD couvre la description du contrôleur de dialogue et de l'interaction abstraite. L'éditeur K-MADe outille la notation [Baron 2006].

Les concepts clefs de la notation sont : les rôles, les utilisateurs, les objets abstraits et concrets, et les évènements. Par conséquent, cette notation permet la description de modèles de tâches, d'objets abstraits et concrets, de rôles et acteurs, d'évènements. Quatre types de tâches sont considérés : abstraite, utilisateur, système et interactive.

Toute tâche est décomposable en sous-tâches. Les règles de composition sont exprimées au niveau de la tâche parent.

3.2 Notations pour la description de tâches multiutilisateurs

3.2.2 GTA

GTA (*Groupware Task Analysis*) [Van der Veer 2000] est une méthode issue de l'ethnographie dédiée principalement à l'analyse de la tâche. Une notation est proposée en support afin de faciliter ce travail d'analyse. Il convient de préciser que GTA est incluse dans une méthode plus globale, la méthode DUTCH [Van der Veer 2002]. DUTCH articule un modèle de tâches système et une description de l'interaction concrète en NUAN (extension de la notation UAN) [Van Welie 1998]. La spécificité de cette méthode est d'exploiter une ontologie reposant sur les concepts clefs de rôle, tâche, objet, agent et évènement. Cette méthode repose sur quatre vues selon quatre représentations distinctes pour décrire l'activité de groupe :

- Un arbre de tâches pour décrire les activités ;
- Un diagramme de classe pour décrire les objets ou artéfacts (type UML) ;
- Un diagramme d'activité ou de séquence pour décrire la dynamique de l'activité ;
- Une représentation de l'environnement physique et culturel sur le plan des usages.

Concernant l'arbre de tâches, le formalisme est simple du fait du nombre réduit de concepts de l'ontologie. Aucun opérateur de composition n'est imposé : le concepteur peut les définir librement. Les tâches peuvent être décorées par des pré- et post-conditions et peuvent également déclencher d'autres tâches (*triggered task*). Ce dernier aspect permet de décrire la dynamique de l'activité. Les tâches élémentaires, c'est-à-dire les feuilles de l'arbre de tâches, sont décrites à l'aide de la notation NUAN.

3.2.3 Cooperative CTT

La notation CCTT (*Cooperative CTT*) est une extension de la notation CTT pour prendre en compte les activités coopératives [Mori 2002]. Cette extension introduit donc un type de tâche supplémentaire, la tâche coopérative, et des arbres de tâches coopératives. Les arbres CTT usuels décrivent alors des tâches individuelles relatives à des rôles métier. L'arbre de tâches coopératives est une composition et structuration de tâches coopératives et de tâches individuelles. Ces dernières sont décorées par le rôle auquel elles sont associées.

CCTT permet de décrire à grain fin l'activité coopérative à l'aide de multiples arbres de tâches individuelles, associés à des rôles métier, et d'un arbre de tâches coopératives décrivant l'activité de groupe, tout en bénéficiant des apports de la notation CTT comme les différents opérateurs logiques et temporels pour exprimer différentes formes de composition.

L'outil CTTE (CTT Environment) [Mori 2002] couvre à la fois CTT et son extension pour les tâches coopératives.

3.2.4 CTML

Reposant sur l'usage d'UML pour la description des objets du domaine, la notation CTML (*Collaborative Task Modeling Language*) se présente comme une extension de

la notation CCTT, dédiée à la modélisation des activités coopératives [Wurdel 2008]. CTML se distingue de CCTT par :

- La proposition d'un cadre pour coupler la description des objets du domaine (diagramme de classe UML) avec la description d'arbres de tâches CCTT ;
- La modélisation des opérateurs temporels offerts par CTT par des machines à état ;
- La proposition d'une description formelle du langage CTML [Wurdel 2008] ;
- L'outillage de la notation combinant un outil d'édition et un outil de simulation, CMTL Editor and Simulator [Wurdel 2008], permettant de simuler et rendre vivant le modèle de tâche par son exécution afin de permettre aux concepteurs de le valider.

Ainsi, la notation CTML définit deux types de modèles que le concepteur doit décrire :

- *Un modèle du domaine* qui décrit les concepts de l'application, dont les rôles utilisateurs, ainsi que les relations entre ces concepts,
- *Un modèle de tâche coopérative* reposant sur les rôles décrits dans le modèle du domaine. Toutefois, les auteurs préconisent de décrire les rôles utilisateurs au sein du diagramme de classe. Pour cela, ils décrivent une classe Rôle dont les autres rôles utilisateur héritent. Les relations entre les concepts de rôles et les concepts de l'application sont décrites avec les relations fournies pour le diagramme de classe UML.

De nouveaux opérateurs sont proposés pour la description du modèle de tâches [Sinnig 2007] :

- *Deux nouveaux opérateurs unaires* : (`#`) qui dénote la capacité de réaliser une tâche donnée plusieurs fois en parallèle ; et (`stop`) qui dénote la terminaison de la tâche mère.
- *Extension de l'opérateur binaire d'alternative* (`[]`) : deux variantes permettent de préciser si un choix est laissé à l'utilisateur (choix déterministe `[]D`) ou réalisé par le système (choix non déterministe `[]N`).

3.2.5 CIAN

S'inscrivant dans le contexte d'une méthode de conception d'application interactive et collaborative (CIAM, *Collaborative Interactive Applications Methodology*) [Molina 2008], la notation CIAN [Molina 2007] vise à décrire au delà des tâches utilisateur l'activité de groupe.. La notation permet de distinguer les activités collaboratives des activités coopératives. Pour cela, plusieurs modèles complémentaires sont proposés, correspondant chacun à une étape du processus de conception. Ces modèles correspondent à des affinements successifs de la description afin d'aboutir à une description très détaillée, par :

1. Un sociogramme permettant de préciser les rôles, les acteurs (humain ou système), et les groupes, et les relations entre ces entités ;
2. Une description des activités de groupe au travers de deux modèles :
    a. Un modèle Inter-Action précisant les grandes classes d'activité sous la forme d'un processus décrivant des enchaînements de type état-transition. Chaque état correspond à une tâche décomposée à gros grain à la façon des machines à états hiérarchiques.

   b. Un modèle de responsabilité précisant pour chaque tâche son type, les objets manipulés, et des pré-conditions. Celui-ci précise les droits d'accès aux objets.
3. Une description détaillée des activités : chaque tâche est décrite par :
   a. Les rôles et acteurs impliqués dans la réalisation de la tâche ainsi que son type et les droits d'accès associés aux rôles ;
   b. Un diagramme de classe UML décrivant les objets manipulés lors de la réalisation de la tâche. Le langage graphique est augmenté de symboles pour marquer les relations entre objets avec les droits d'accès afférents ;
   c. Un arbre de tâches CTT augmenté des notions de droits d'accès.

3.2.6 TOUCHE

La méthode TOUCHE [Penichet 2010] pour la conception des systèmes interactifs multiutilisateurs (*Task-Oriented and User-Centred Process Model for Developing Interfaces for Human–Computer–Human Environments*) se concentre sur la description des interactions entre acteurs et rôles. Cette méthode centrée rôle vise à fournir des outils au concepteur pour comprendre et modéliser finement une organisation en termes de rôle et de structure, ainsi que les relations collaboratives entre acteurs de cette organisation, en lien avec les tâches associées aux rôles. Deux classes de modèles sont employées pour décrire :

- *La structure de l'application* : outre l'usage de diagrammes de classe UML pour préciser la structure de l'application, la méthode introduit le diagramme de structure organisationnelle (OSD) qui précise la structure d'une organisation composée de groupes, de rôles et d'acteurs ainsi que les relations entre rôles et acteurs.
- *Le comportement de l'application* : le comportement est décrit par des arbres de tâches individuelles (TD) et un diagramme de co-interactions (CD). La description des arbres de tâches (TD) repose sur la notation CTT. Toutefois, la méthode permet l'emploi d'autres notations comme GTA. Le diagramme co-interactions (CD) est introduit par la méthode et précise les interactions par des relations entre rôles. Chaque relation précise les effets de l'interaction en terme d'action et de réception. Contrairement à une structure de nature hiérarchique, la structure est une composition de graphes.

Le diagramme de structure organisationnelle est très proche du sociogramme de CIAN, aussi bien en terme de concepts qu'en terme de représentation. Son pouvoir d'expression pour la description des rôles est néanmoins moins puissant qu'un sociogramme CIAN, de par l'absence de la relation d'héritage. En revanche, le diagramme de co-interactions permet d'expliciter de manière synthétique le réseau de collaborations entre les intervenants d'une situation, ce que les autres notations présentées jusqu'ici proposent de décrire au sein d'un modèle de tâches coopératives.

3.2.7 MABTA

Dédiée à l'analyse de la tâche et des besoins, la notation MABTA (*Multiple Aspect Based Task Analysis*) [Lim 2004] repose sur quatre modèles pour couvrir différentes facettes de l'activité collaborative :

- *Modèle de rôles* : exprime la répartition des rôles selon une structure hiérarchique entre utilisateurs.

- *Modèle de tâches de groupe* : permet de définir des classes de tâches de groupe associées à des rôles. Les relations entre ces tâches expriment une conséquence, similaire au *trigger* introduit par la notation GTA. Trois types de tâches sont considérés : coordination, décision et mono-utilisateur.
- *Modèle de tâches* : ce modèle est une version détaillée du modèle de tâches de groupe : chaque tâche de groupe est exprimée par une liste de tâches élémentaires. La composition de ces tâches élémentaires est exprimée par un plan d'action, approche développée par la notation HTA, reposant sur l'usage d'opérateurs usuels comme la séquence ou l'alternative.
- *Modèle d'interface abstraite* : ce modèle permet d'expliciter une structuration de l'espace graphique d'interaction tout en exprimant des liens avec le modèle de tâches.

3.2.8 HAMSTERS

La notation HAMSTERS (*Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems*) [Martinie 2014] est une notation pour la modélisation des tâches utilisateurs. Elle emprunte aux notations existantes des concepts comme la structuration hiérarchique ou les opérateurs LOTOS (voir notation CTT). Néanmoins la notation HAMSTERS propose des types de tâches de granularité plus fine que CTT, comme la distinction entre tâche d'interaction en entrée ou en sortie (que l'on trouve également dans d'autres notations comme la notation COMM (voir section 3.3.5). Parmi les concepts véhiculés par la notation, la notation HAMSTERS promeut des mécanismes pour faciliter la modularité et la réutilisation de sous-arbres (*subroutine* et *copy task*) et rend explicite les flots de données entre tâches (notion de port d'entrée et de sortie) ainsi que les objets manipulés par une tâche. De plus, la notation fournit des éléments pour prendre en compte l'erreur humaine. Enfin, la notation est outillée par un éditeur complet permettant l'édition de tâches ainsi que la simulation de ces tâches.

Cette notation s'inscrit de plus dans une méthode pour la conception de systèmes interactifs critiques. Cette méthode adosse la notation HAMSTERS avec la notation ICO (voir section 3.3.3), cette dernière permettant la spécification formelle du comportement du système interactif en cours de conception. Du point de vue de la tâche, la notation ICO permet une description des tâches élémentaires concrètes.

L'aspect multiutilisateur est couvert par l'inclusion de plusieurs types de tâches collaboratives : tâche de groupe, tâche de groupe abstraite, tâche de groupe interactive, tâche de groupe système. Ces tâches peuvent être elles-mêmes composées de tâches de groupe ainsi que de tâches individuelles. De plus, la notation inclut des décorations permettant de préciser la nature de la tâche au regard des dimensions espace-temps (distant/local et synchrone/asynchrone). La notation permet également de préciser la couverture fonctionnelle au regard du modèle du trèfle (communication, coordination, production).

3.2.9 COMM

La notation COMM (*COllaborative and MultiModal*) [Jourde 2010] est une notation à base d'arbre de tâches, dans l'esprit de la notation CTT dont elle étend les opérateurs. COMM permet la description conjointe de tâches multiutilisateurs et de l'interaction concrète multimodale pour les tâches élémentaires. La particularité est de permettre la description des tâches individuelles et de groupe au sein d'un unique arbre de tâches en s'appuyant sur les types de tâches suivants :

- Deux tâches système : calcul (interne : changement d'état) et présentation ;
- Trois tâches individuelles : utilisateur (interne : cognitive), action et interaction (qui combine la tâche système présentation avec la tâche utilisateur action).
- Trois tâches de groupe : groupe (interne : décision), action de groupe et interaction de groupe (qui combine la tâche système présentation avec la tâche utilisateur action).

Chaque tâche peut être décorée avec un ou plusieurs rôles métiers. De plus, la notation COMM permet de préciser un nombre d'acteurs parmi un ensemble de rôles pour réaliser une tâche de groupe ainsi que des règles de composition et factorisation en fonction des types de tâches et de la décoration par les rôles. Elle couvre à la fois la description de tâche abstraite, qui souvent relève de tâches de groupe, et la description de l'interaction concrète par l'introduction du type de tâche modale pour définir une modalité d'interaction comme étant un couple `<dispositif, langage d'interaction>`.

L'introduction de la notion de rôle interactif, complémentaire au concept de rôle métier, permet de décrire des tâches collaboratives. Aussi, le rôle interactif permet d'expliciter l'allocation des rôles métiers aux utilisateurs du point de vue du dispositif interactif. En effet, contrairement à une tâche coopérative, il n'est pas toujours possible de décrire a priori l'allocation des tâches en associant un rôle métier aux utilisateurs : souvent cette allocation est effectuée à l'exécution. Ainsi, le rôle interactif traduit ce caractère dynamique de l'interaction explicitant le tissage entre tâches relevant de l'interaction concrète avec les rôles métiers.

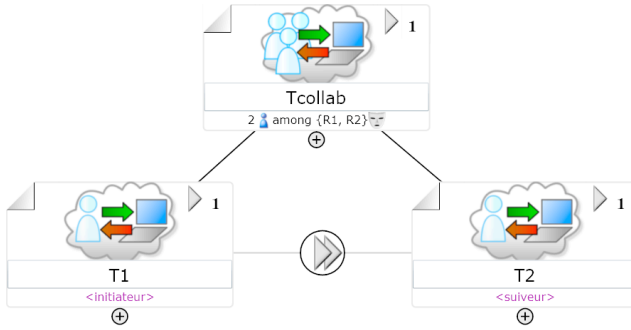


Fig. 3. Exemple de tâche collaborative reposant sur la notion de rôle interactif, spécifiée avec la notation COMM.

Pour illustrer ce concept de rôle interactif, considérons une tâche collaborative, nommée Tcollab, sans allocation prédéfinie des rôles métiers aux tâches, consistant à exécuter en séquence une tâche T1 puis une tâche T2, accomplies par deux rôles métiers distincts, R1 et R2. L'opportunité est laissée aux acteurs de se coordonner et d'accomplir cette tâche collaborative : l'allocation des tâches aux rôles est réalisée au moment de l'interaction effective des utilisateurs avec le système, en fonction des contraintes induites par le dispositif interactionnel. Pour la spécifier à l'aide de la notation COMM : comme le montre la Figure 3, la tâche collaborative Tcollab est associée à deux rôles métiers (R1 et R2), et est composée de deux tâches individuelles accomplies en séquence, T1 et T2. La décoration de la tâche collaborative Tcollab exprime la contrainte que deux rôles métiers distincts, parmi une liste de rôles {R1, R2}, sont nécessaires pour être accomplies. De plus, les deux sous-tâches T1 et T2 étant des tâches individuelles, un seul rôle métier peut y être associé. Aussi, l'imprévisibilité de l'allocation des tâches T1 et T2 aux rôles R1 et R2 est spécifiée par la décoration des

tâches individuelles par deux rôles interactifs distincts, respectivement <initiateur> et <suiveur>. Ainsi, au moment de l'interaction, un seul des deux rôles métiers R1 et R2 sera <initiateur>, et l'autre <suiveur>.

La notation COMM a été employée pour décrire les tâches d'un prototype d'IHM de poste de commande de drones. Ce projet a permis de mettre à l'épreuve la notation COMM et de valider son passage à l'échelle ainsi que son pouvoir expressif : plus de 180 tâches ont été décrites pour des arbres constitués de branches s'étalant sur 12 niveaux de profondeur. En particulier, la notion de rôle interactif permet l'expression de la délégation de tâches relative au partage d'autorité.

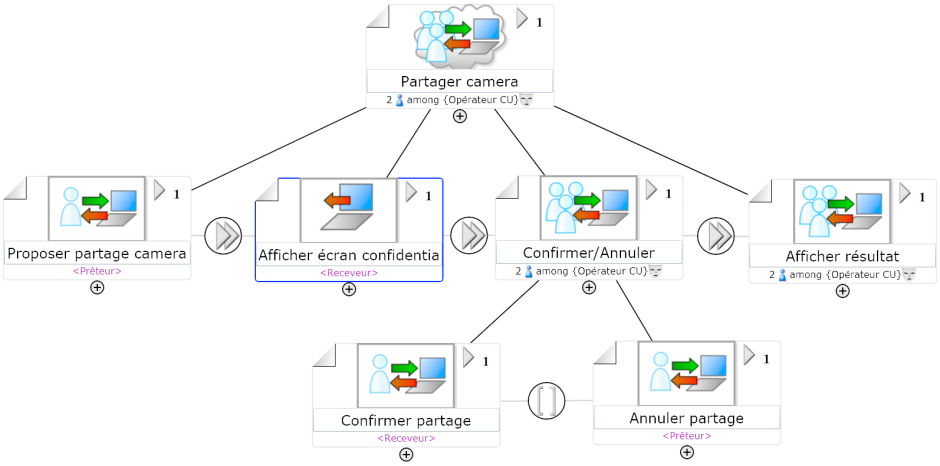


Fig. 4. Tâche de délégation de contrôle de charge utile entre opérateurs, exprimée à l'aide de deux rôles interactifs.

La Figure 4 illustre un exemple concret tiré de ce projet, décrit à l'aide de la notation COMM. Le contrôle d'un drone implique plusieurs opérateurs, c'est-à-dire des rôles métiers : un opérateur vecteur (VE) pour le contrôle de la trajectoire ; un ou plusieurs opérateurs charge utile (CU) pour le contrôle des dispositifs embarqués comme une caméra 360˚. Le concept de rôle interactif a été utilisé pour spécifier une tâche de partage de charge utile entre deux opérateurs CU. Comme le montre la Figure 4, la tâche collaborative consiste à déléguer le contrôle d'une caméra 360˚ entre deux opérateurs CU : elle illustre la phase de partage à l'aide des deux rôles interactifs <Prêteur>, qui délègue le contrôle, et <Receveur>, qui en prend le contrôle.

Outre le concept de rôle interactif, la notation COMM propose un éventail étendu de types de tâches [Jourde et coll., 2010 ; Jourde, 2011] :

- Deux tâches système : calcul (interne : changement d'état) et présentation (par exemple, la tâche « Afficher écran » de la Figure 4) ;
- Trois tâches individuelles : utilisateur (interne : cognitive), action et interaction (qui combine la tâche système présentation avec la tâche utilisateur action), comme la tâche « Annuler partage » à la Figure 4 ;
- Trois tâches de groupe : groupe (interne : décision), action de groupe et interaction de groupe (qui combine la tâche système présentation avec la tâche utilisateur action), comme la tâche « Partager Caméra » à la Figure 4.

La notation préconise aussi un ensemble de règles de composition et de factorisation selon les types de tâches [Jourde, 2011].

En outre, la notation COMM est outillée par l'éditeur e-COMM [Jourde 2010 e-COMM], disponible sous la forme d'une application web. COMM et e-COMM sont complètement décrits et illustrés par des exemples de description de systèmes interactifs dans [Jourde 2010].

3.3 Notations pour la description de tâches élémentaires concrètes multimodales

Pour les tâches élémentaires concrètes ou actions mono/multimodales de l'utilisateur nous considérons les notations de description de l'interaction concrète (comme NUAN pour décrire les tâches élémentaires concrètes dans la notation GTA – voir section 3.2.2). Dans cette section, nous considérons les notations spécifiques à la multimodalité, que ce soit pour décrire les actions utilisateur multimodales ou pour décrire le système autorisant ces actions multimodales.

3.3.1 ICOM

La notation ICOM (Input Configuration Model) [Dragicevic 2004] permet de décrire les entrées du système sous la forme d'un diagramme de flot de données, composé de modules interconnectés. ICOM permet de décrire l'interaction concrète Post-WIMP et multimodale pour la réalisation de tâches données d'un système interactif. La notation ICOM est opérationnalisée au sein de l'environnement ICON, qui permet de décrire des diagrammes ou configurations d'entrée selon la notation ICOM mais également de les exécuter.

Pour décrire une configuration d'entrées, la notation ICOM utilise des entités de base appelées dispositif. Les entités disposent d'entrées et de sorties qui sont définies sous la forme de ports d'entrée et de sortie. Ces ports sont définis par un type de base tel que le type booléen ou le type entier, ou sont composés pour former un type complexe. Les dispositifs peuvent être de trois types : des dispositifs d'interaction tels qu'une souris, des dispositifs d'application qui correspondent aux tâches élémentaires proposées par le système et les dispositifs d'adaptation qui permettent de transformer des informations provenant des dispositifs d'interaction dans une forme compréhensible par les dispositifs d'application. Une configuration d'entrées est ainsi définie par un ensemble de dispositifs interconnectés.

La notation ICOM permet donc de décrire les traitements du système pour l'interaction concrète multimodale en entrée sous la forme d'un flot de données entre les dispositifs physiques d'interaction et les tâches élémentaires concrètes.

3.3.2 ICARE

Comme ICOM, la notation ICARE (Interaction, Complémentarité, Assignation, Redondance, Equivalence) [Bouchet 2004] permet de décrire des assemblages de composants au sein d'un diagramme de flot de données pour décrire les traitements du système pour l'interaction multimodale en entrée. ICARE repose sur la définition d'une modalité comme étant un couple `<dispositif, langage d'interaction>`, et propose ainsi de décrire une modalité sous la forme de deux composants : un composant dispositif et un composant langage d'interaction. Des composants de combinaison sont proposés pour exprimer la composition de modalités selon les propriétés CARE [Coutaz 1995] : complémentarité, redondance, et redondance/équivalence.

Avec ICARE, pour chaque tâche élémentaire concrète d'une application interactive, une description sous forme d'un assemblage de composants est produite. Au sein de l'outil de prototypage ICARE, les diagrammes font apparaître des composants décrits

par de nombreuses propriétés telles que la précision, la stabilité ou encore le mode de communication. Les assemblages produits peuvent alors être exécutés, et les combinaisons de modalités testées, afin de valider l'utilisation de ces modalités pour une tâche donnée.

Les principes de description d'ICARE ont été repris avec un autre modèle à composants permettant l'intégration de composants logiciels variés au sein de la plate-forme logicielle OpenInterface [Serrano 2008].

3.3.3 ICO

ICO (*Interactive Cooperative Objects*) [Navarre 2009] est un langage formel reposant sur les réseaux de Petri pour décrire les traitements système pour l'interaction utilisateur. Les différentes évolutions de ce langage couvrent désormais la description des interfaces multimodales, WIMP et Post-WIMP [Navarre 2009]. Un réseau de Petri est un graphe constitué d'arcs orientés pouvant relier deux types de nœuds : des états (représentés par une ellipse) contenant des jetons associés à des valeurs ; des transitions (représentées par un rectangle) explicitant des changements d'états ou des actions. Un objet ICO repose sur une approche par objet des réseaux de Petri permettant une description d'objets complexes et la description complète d'un comportement d'un objet en réaction à des stimuli externes. Par rapport aux Objets Coopératifs (CO), le langage ICO ajoute la notion d'évènement utilisateur et permet de rendre explicite les différents liens via ce mécanisme d'évènement entre les composants relatifs à l'interaction (concrète et abstraite), le dialogue et le noyau fonctionnel (que l'on peut associer à un objet CO). Le langage est outillé par l'application PetShop.

Le langage ICO, du fait des propriétés des réseaux de Petri, permet la description fine du traitement système de l'interaction multimodale grâce à un réseau combinant des évènements produits par plusieurs interacteurs au sein d'un unique réseau de Petri disposant de plusieurs jetons. Un tel modèle exprime donc un mécanisme de fusion et permet de préciser les contraintes temporelles régissant la fusion multimodale.

3.3.4 Dynamo-Aid

Dynamo-Aid [Clerckx 2007] est une méthode de conception de prototypes d'application interactive sensible au contexte d'usage, issue de recherche en informatique pervasive. Les mécanismes d'adaptation considérés par la méthode sont la capacité du système interactif à répartir son IHM sur plusieurs dispositifs et la capacité à exploiter des modalités ou combinaisons de modalités variées. En particulier, l'expression de combinaisons de modalités repose explicitement sur l'usage des propriétés CARE [Coutaz 1995].

Plusieurs notations sont proposées en support à la méthode Dynamo-Aid afin de permettre de décrire un système multimodal à l'aide de modèles de : tâche, dialogue, contexte et présentation. L'intégration de tous ces modèles est réalisée au travers d'un modèle complémentaire, le modèle de l'interface qui exprime les interrelations.

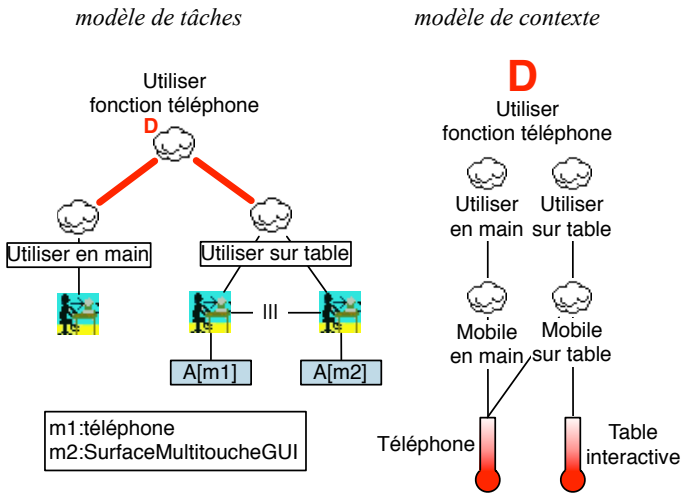


Fig. 5. Description d'une tâche selon des contextes d'usage différents (ici en fonction de la plateforme) : modèle de tâche (à gauche) et modèle de contexte (à droite).

Les spécificités de ces modèles sont :

- *Modèle de tâches* : une tâche est décrite à l'aide de la notation CTT étendue de la notion de modalité (assimilée à celle de dispositif) pour préciser la nature des tâches d'interaction élémentaire. Aussi, une tâche d'interaction est associée à une modalité (vocale, gestuelle, etc.) ou une combinaison de modalités. Une combinaison est exprimée par une des quatre propriétés CARE qui agit comme un opérateur liant plusieurs modalités. Comme le montre la Figure 5, c'est la notation CTT qui est utilisée pour décrire la tâche « Utiliser sur table ». La notation Dymano-Aid permet une description de l'interaction multimodale par spécialisation de la tâche élémentaire d'interaction. Dans cet exemple, il s'agit d'une assignation de l'usage parallèle de deux modalités m1 et m2, associées respectivement au dispositif de type téléphone et au dispositif de table interactive multitouche. De plus, la notation introduit la notion de tâche de décision qui, selon le contexte, active une des sous-branches de la tâche de décision (voir modèle de contexte de la Figure 5). Dans l'exemple de la Figure 5, la tâche de décision a deux branches décrivant un usage du téléphone en main et un usage du téléphone posé sur la table.
- *Modèle de dialogues* : la notation prescrit la description du dialogue à l'aide de diagrammes état-transition, un état représentant un état de l'interface utilisateur. Les transitions marquent un changement de l'interface, suite à une action de l'utilisateur notamment.
- *Modèle de contextes* : ce modèle repose sur deux classes d'objets du contexte, concrets et abstraits. Les objets concrets sont associés à des dispositifs physiques (des capteurs) capables de saisir des informations du contexte d'usage comme la température. Ces objets concrets sont associés à des objets abstraits du contexte reflétant une caractéristique du contexte (il fait chaud / il fait froid). Le contexte d'usage inclut également la plateforme, donc les types de dispositifs employés. Dans l'exemple de la figure 5, deux contextes dépendant des dispositifs sont décrits : un usage du téléphone en main et un usage du téléphone posé sur la table interactive. Pour le second cas, le contexte dépend des deux dispositifs, téléphone et table interactive.

Ensuite, les objets abstraits sont reliés au modèle de tâche et de dialogue en exprimant des contraintes sur leur disponibilité dans l'arbre (s'il fait chaud, rendre possible l'usage du ventilateur). Précisément, comme le montre la Figure 5, la notation CTT est augmentée du type de tâche de Décision signifiant des alternatives dont le choix repose sur l'état du contexte, exposé par les objets abstraits du contexte, et des conditions posées sur cet état. De même, un modèle de dialogue peut être composé de grappes d'états correspondant à différentes versions du dialogue pour des contextes variés : en effet les transitions entre ces grappes sont des transitions dont les conditions sont exprimées par des contraintes dépendantes du contexte via les objets abstraits du contexte.

- *Modèle de l'application* : ce modèle est une représentation du noyau fonctionnel. Les changements d'états internes (service indisponible par exemple) sont perçus également comme étant un changement du contexte influençant les modèles de tâche et de dialogue.
- *Modèle de présentations* : ce modèle correspond à l'interface finale générée à partir des modèles précédents.

3.3.5 COMM

Outre la description des tâches multiutilisateur (paragraphe 3.2.8), la notation COMM permet la description de l'interaction concrète pour les tâches élémentaires et, en particulier, l'interaction multimodale. La notion de tâche modale constitue la base pour exprimer la multimodalité étant donné qu'elle permet de préciser une modalité par le couple `<dispositif, langage d'interaction>`.

La description d'une interaction multimodale se concrétise en terme de combinaison de tâches modales par un sous-arbre. Les propriétés CARE s'expriment de deux façons : par combinaison de tâches et à l'aide des opérateurs temporels.

En terme de combinaison de tâches, l'assignation exprime l'absence de choix : une tâche élémentaire est simplement une tâche modale. Il n' y a pas de contrainte temporelle. L'équivalence exprime une situation de choix entre plusieurs modalités : une tâche élémentaire est alors une composition de tâches modales reliées par un opérateur d'alternative, avec ou sans contrainte temporelle. La complémentarité exprime l'usage conjoint de plusieurs modalités (c'est-à-dire qu'aucune modalité ne peut être utilisée individuellement pour accomplir la tâche) dans une fenêtre temporelle : une tâche élémentaire est alors une combinaison d'au moins deux tâches modales, reliées par un opérateur temporel. La redondance exprime l'usage de modalités pour exprimer la même information soit en parallèle soit en séquence, dans une fenêtre temporelle : une tâche élémentaire est alors la composition d'au moins deux tâches modales reliées par un opérateur temporel.

En outre, la notation COMM propose un éventail étendu des opérateurs temporels LOTOS présents dans la notation CTT par des opérateurs précisant des agencements temporels à grain plus fin sur la base des relations de Allen [Allen 1983] appliquées à la multimodalité [Vernier 2000 ; Serrano 2009]. Ainsi, l'opérateur exprimant le parallélisme (`|||`) est affiné en trois opérateurs exprimant : la concomitance, la coïncidence, et le parallélisme dans l'utilisation des modalités. De même, l'opérateur de séquence est affiné en deux opérateurs : l'anachronisme et la séquence.

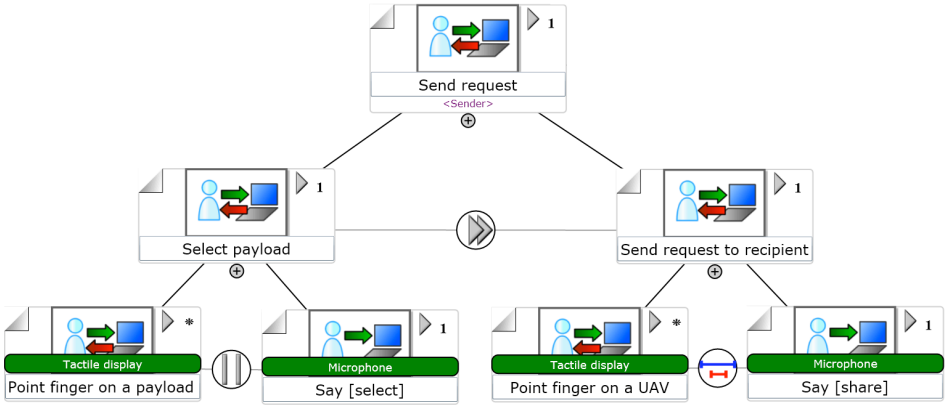


Fig. 6. Tâche multimodale de partage de contrôle de charge utile d'un drone.

L'exemple de description de la Figure 6 est un cas concret qui spécifie le transfert du contrôle de charge utile d'un drone entre deux opérateurs, et illustre deux tâches multimodales (tâches « Select payload » et « Send request to recipient ») composées chacune de deux tâches modales complémentaires. Ainsi, afin de réaliser la tâche « Select payload », l'utilisateur doit poser son doigt sur l'écran tactile tout en formulant oralement l'ordre « select ». L'opérateur de parallélisme reliant ces deux modalités exprime donc une contrainte temporelle entre celles-ci, à savoir une utilisation concurrente des deux modalités. Néanmoins cette contrainte temporelle n'est pas assez précise pour exprimer une exécution concomitante. Aussi, sur la base des opérateurs temporels étendus, la seconde tâche d'interaction concrète « Send request to recipient » est plus précise que la tâche « Select payload » : les deux tâches modales « Point Finger on UAV » et « Say [share] » sont reliées par l'opérateur de concomitance. Cela signifie que l'utilisation de l'une des deux modalités démarre avant l'autre et que son utilisation se terminera avant. L'usage de tels opérateurs dans une description constitue donc une seconde brique pour spécifier l'interaction multimodale.

Table I. Synthèse relative aux critères généraux et à la description des tâches utilisateur.

Notation	Champ disciplinaire	Couverture [ARCH]	Représentations	Modèle	Instrumentation	Structuration	Composition	Objets
CTT	IHM	[CD, IA]	1: Graphique	Méta-modèle	CTTE	Arbre	op. LOTOS	textuel
SAMANTA	IHM	[CD, IA]	1: Graphique	Méta-modèle	SAMANTA	Arbre, Graphe	op. LOTOS	--
K-MAD	Psycho, IHM	[CD, IA/C]	4: Graphique, tabulaire	Méta-modèle	K-MADe	Arbre	opérateurs temporels	collections
GTA	Psycho, ethno, TCAO	[CD, IA/C]	4: Graphique, texte, tabulaire	Onto-logie	EUTERPE	Arbre, pseudo digrammes UML	oui (indéfini)	pseudo diag. classe
Coop. CTT	TCAO	[CD, IA]	1: Graphique	Méta-modèle	CTTE	Arbre	op. LOTOS	textuel
CTML	IHM	[CD, IA]	1: Graphique	langage formel	CTML editor	Arbre+UML	op. LOTOS étendus	UML
CIAN	IHM, TCAO	[NF, CD, IA/C]	5: Graphique, tabulaire	Méta-modèle	CIAT (Eclipse)	Arbres+UML	op. LOTOS	UML
TOUCHE	IHM	[CD, IA]	4: Graphique, texte, tabulaire	Méta-modèle	TOUCHE CASE	Arbre, graphes	op. LOTOS	UML
MABTA	Psycho, TCAO	[CD, IA]	4: Graphique, sketch	Méta-modèle	--	Graphes	Plans HTA	--
HAMSTERS	GL,IHM	[(NF), CD,IA, (IC)]	1: Graphique	Méta-modèle	HASMTERS (+ PetShop)	Arbre	op. LOTOS	UML
ICOM	IHM	[IC]	1: Graphique	--	outil ICON	Graphe	dispositif / connexion	--
ICARE	IHM	[IC]	1: Graphique	--	outil ICARE de prototypage	Graphe	prp. CARE	--
ICO	GL,IHM	[NF, CD, IA/C]	1: Graphique	Petri	PetShop	Réseaux de Petri	transitions / jetons	classes
Dynamo-Aid	IHM	[CD, IA/C]	4: Graphique	Méta-modèle	Dynamo-Aid tool	Arbre, graphe	op. LOTOS	Interactive abstract obj.
COMM	IHM, TCAO	[CD, IA/C]	3: Graphique	CTT	e-COMM	Arbre+UML	op. LOTOS étendus	UML

Table II. Synthèse relative à la description des tâches multiutilisateur et interaction multimodale.

Notation	Interaction multiutilisateur					Interaction multimodale			
	Rôle	Agent	Tâche	Objet	Evènement	Modalité	Dispositif	Langage	Composition
CTT	--	--	--	Objet	--	Non	Non	Non	--
SAMANTA	--	--	--	--	--	Non	Non	Non	--
K-MAD	Rôle	Acteur	--	Objet	Evènement	Non	Oui	Non	--
GTA	Rôle	Agent	Tâche	Objet	Evènement	Non	(Oui : NUAN)	(Oui : NUAN)	--
Coop. CTT	Rôle	Utilisateur	Individuelle, coopérative	Objet	--	Non	Non	Non	--
CTML	Rôle	Utilisateur	Individuelle, coopérative	Objet	Effet	Non	Non	Non	--
CIAN	Rôle	Utilisateur	coopérative, collaborative	Objet, donnée partagée	Notification	Non	Non	Non	--
TOUCHE	Rôle	Acteur, utilisateur	groupe	Objet	--	Non	Non	Non	--
MABTA	Rôle	Acteur, groupe	But, action, tâche	Ressource	--	Non	Non	Non	--
HAMSTERS	Rôle	Acteur	Individuelle, coopérative	Objet	--	Non	Non	Non	--
ICOM	--	--	--	--	--	Non	Oui	Non	dispositif d'adaptation
ICARE	--	--	--	--	--	Non	Oui	Oui	prp. CARE
ICO	--	--	--	--	--	Non	Oui	Non	Transitions / jetons
Dynamo-Aid	--	--	--	--	--	Oui	Oui	Non	prp. CARE
COMM	Rôle métier	Rôle interactif	coopérative, collaborative	Objet	Déclencheur	Non	Oui	Oui	prp. CARE, rel. Allen

4. SYNTHÈSE

Cette partie présente une synthèse des différentes notations décrites dans la partie précédente selon les critères identifiés et détaillés en partie 2. Deux tableaux (Table I et Table II) récapitulent les points clés pour chaque critère. Notamment, dans la Table I, la colonne "Représentation" indique le nombre de représentations offertes par la notation suivi des types de représentations. Dans la colonne "Composition", les abréviations "op." et "prp." signifient respectivement opérateur et propriété. Dans la seconde table (Table II), les colonnes "Modalité", "Dispositif" et "Langage" indiquent par oui ou non si la notion est explicite au sein de chaque notation. Dans la colonne "Composition" (Table II), les abréviations "op." et "rel." signifient respectivement opérateur et relation.

4.1 Caractéristiques générales et liées à la description des tâches utilisateur

La majorité des notations sont principalement dédiées aux deux étapes du processus de développement logiciel : analyse des besoins et description de l'interaction. Les notations ICOM, ICARE et ICO font exception puisqu'elles relèvent principalement de la conception logicielle : Nous les avons présentées pour étudier la description des tâches élémentaires multimodales ou interaction multimodale concrète, que permet le traitement système décrit par ces notations. Tandis que ICOM et ICARE reposent sur des composants logiciels, le langage ICO repose sur les réseaux de Petri et permet ainsi de décrire finement le traitement système de l'interaction. De plus ICO couvre l'ensemble de ARCH contrairement à ICOM et ICARE dédiés à l'interaction concrète.

Toujours du point de vue des niveaux d'abstraction de ARCH, un grand nombre de notations sont employées pour décrire les niveaux dialogue et interaction abstraite. Certaines notations couvrent également le niveau interaction concrète (K-MAD, GTA avec NUAN, CIAN, Dynamo-Aid, HAMSTERS/ICO et COMM).

Un large ensemble de ces notations décrivent les tâches utilisateur sous la forme graphique, précisément par des arbres de tâches système et par des diagrammes de classe pour préciser les objets du domaine manipulés lors de l'accomplissement des tâches utilisateur. Le diagramme de classe UML est souvent privilégié comme notation complémentaire pour exprimer les objets du domaine. Toutefois, la notation GTA propose un pseudo langage graphique pour exprimer des classes d'objets tandis que les notations TOUCHE/CIAN et MABTA/ICARE impliquent une représentation respectivement de type *workflow* et de type flots de données.

Pour exprimer la composition des tâches, les notations à base d'arbres de tâches reposent sur l'usage d'opérateurs logiques et temporels, principalement ceux dérivés de LOTOS comme c'est le cas pour la notation CTT. Comme le souligne le schéma de la Figure 2, ceci s'explique par le fait que de nombreuses notations reposent sur CTT (par exemple CTML, HAMSTERS, SAMANTA ou Dynamo-Aid). Pour certaines notations comme CTML, HAMSTERS, et COMM, ce jeu d'opérateurs est même étendu pour une description à grain plus fin, notamment de l'interaction multimodale (COMM et CTML). Dans le cas de la notation GTA, la notion d'opérateur est présente pour exprimer la composition de tâches mais la définition d'opérateurs est laissée au concepteur. Les opérateurs temporels élémentaires comme la séquence ou les opérateurs logiques comme l'alternative sont proposés par toutes les notations à base d'arbres de tâches. La notation MABTA repose sur le concept de plan, hérité de la notation HTA, pour décrire l'enchaînement de tâches. Dans le cas du langage ICO, la composition s'exprime par le biais d'un ensemble de réseaux de Petri reliés par le principe d'envoi

d'évènements pour une description du traitement système de l'interaction concrète liée à une description conjointe de la tâche à l'aide de la notation HAMSTERS.

Outre ICOM et ICARE, les notations reposent sur un modèle exprimé soit par un méta-modèle, soit par un langage formel comme les réseaux de Petri, soit spécifique comme le langage CTML.

4.2 Caractéristiques liées à la description des tâches multiutilisateurs et tâches élémentaires concrètes multimodales

Deux notations permettent la description des tâches multiutilisateurs et multimodales : COMM et, dans une certaine mesure, Dynamo-Aid en considérant la version étendue aux tâches coopératives de la notation CTT, c'est-à-dire CCTT. C'est également le cas de la notation HAMSTERS qui a été étendue par, entre autre, l'introduction de nouveaux types de tâches collaboratives. Les autres notations sont clairement spécialisées, comme CCTT (Cooperative CTT) et K-MAD ciblant la description des tâches multiutilisateurs, ou comme ICOM, ICARE et ICO ciblant le traitement système de l'interaction multimodale (par rapport à HAMSTERS, ICO est préconisée pour décrire le traitement système de l'interaction concrète).

La notation SAMANTA permet indirectement de décrire des tâches entre utilisateurs au travers de la notion de Situation Awareness, reflétant la connaissance commune d'un contexte de l'interaction pour réaliser une tâche. Les autres notations visant la description des tâches multiutilisateurs reposent sur un socle commun de concepts : rôle, tâche individuelle et coopérative, groupe, acteur, objet (partagé ou non), et évènement. Les notations MABTA, CIAN et TOUCHE reposent sur un éventail plus large de concepts car ces notations visent une description fine des groupes et des structures organisationnelles, notamment par le biais de représentations complémentaires à base de sociogrammes. En termes de tâches multiutilisateurs, les notations CIAN et COMM permettent une description fine en distinguant les deux classes d'activité de groupe : collaborative et coopérative. De plus, tout comme HAMSTERS, la notation COMM étend la notation CCTT en proposant un éventail plus large de types de tâches multiutilisateurs : tâche de groupe, tâche d'action de groupe et tâche d'interaction de groupe. Enfin, la notation COMM introduit la notion de rôle interactif permettant l'expression de tâches collaboratives. Ce concept permet de maintenir un lien étroit entre la description des tâches abstraites et des tâches concrètes élémentaires.

Sur le plan de la description de l'interaction multimodale, les notations Dynamo-Aid, ICOM, ICARE et COMM explicitent les concepts de modalité, de langage ou de dispositif. La notation ICOM considère seulement l'aspect dispositif. Au contraire, l'aspect langage est explicite dans la notation COMM au travers du concept de tâche modale. Dans une certaine mesure, notons que la notation GTA, via la notation NUAN, permet la description du langage d'interaction associé à une modalité. Pour exprimer une interaction multimodale, les notations Dynamo-Aid, ICOM, ICARE et COMM explicitent les propriétés CARE pour décrire la combinaison de modalités selon différentes formes :

- par l'introduction de conditions permettant de relier plusieurs modalités associées à une tâche sous la forme d'une décoration (Dynamo-Aid) ;
- par la notion de dispositif d'adaptation (ICOM) ;
- par le principe de composant associant des composants encapsulant une modalité (ICARE) ;

- par l'introduction du concept de tâche modale et par l'extension du jeu d'opérateurs LOTOS à l'aide des relations de Allen pour exprimer finement des combinaisons de modalités selon les propriétés CARE (COMM).

Pour les autres notations, l'expression de la multimodalité est possible mais n'est pas prise en compte explicitement.

Le langage ICO permet une description précise de l'interaction utilisateur en général et multimodale en particulier, notamment du point de vue de la fusion des évènements multimodaux. Il doit être cependant possible d'exprimer des patrons relatifs aux propriétés CARE captant des descriptions récurrentes à l'aide de réseaux de Petri.

4.3 Illustration et analyse des notations Dynamo-Aid et COMM

L'étude comparative de notations existantes met donc en évidence que deux notations, Dynamo-Aid [Clerckx 2007] et COMM [Jourde 2010], véhiculent explicitement des concepts pour la description des tâches multiutilisateurs et tâches concrètes multimodales. Si la notation CCTT est utilisée pour la première, il est alors possible de décrire les tâches multiutilisateurs conjointement aux tâches élémentaires concrètes multimodales.

Afin d'illustrer et comparer Dynamo-Aid et COMM , nous considérons un exemple de tâche collaborative et de tâche concrète multimodale tiré et adapté de [Tse 2008] (Figure 7). Précisément, nous illustrons au travers de cet exemple la description des tâches abstraites et concrètes multiutilisateurs et multimodales. Nous concluons cette partie avec des éléments d'analyse comparative.

Fig. 7. Système collaboratif et multimodal Warcraft III [Tse 2008] (Image tirée de https://sites.google.com/site/edwardhtse2/areasofcreativity (D.R.).

Dans le contexte du jeu Warcraft III, la tâche collaborative retenue pour illustrer l'usage de ces deux notations consiste à sélectionner un soldat pour ensuite lui indiquer une destination afin de le déplacer sur le plateau de jeu. Cette tâche est composée de deux sous-tâches d'interaction individuelle, exécutées en séquence : choisir un soldat puis choisir une destination. Deux rôles sont considérés pour accomplir cette tâche : un rôle associé à la sous-tâche de sélection du soldat (R sélection) et un rôle associé à la sous-tâche d'indication de destination (R destination).

À l'aide de la notation CCTT pour exprimer la tâche collaborative, la tâche conjointe pour déplacer un soldat (Figure 8) est décrite avec la notation Dynamo-Aid par une tâche coopérative : celle-ci est composée de trois sous-tâches dont deux sous-tâches abstraites de décision (D), chacune associée à un des deux rôles. Les deux tâches abstraites de décision sont deux tâches d'interaction individuelle (Figure 9) décrivant

une interaction multimodale à l'aide de deux modalités, m1 et m2, associées respectivement à deux dispositifs d'interaction en entrée : la surface multitouche et un microphone. Pour la première sous-tâche « Choisir un soldat », l'usage conjoint des deux modalités est décrit explicitement à l'aide de l'opérateur de complémentarité (CARE) associant les modalités m1 et m2 (C[m1, m2] à la Figure 9). Pour la seconde sous-tâche « Choisir destination », nous montrons qu'il est également possible d'exprimer la complémentarité (CARE) à l'aide de deux sous-tâches exécutées en parallèle, chacune associée à une des deux modalités par assignation (CARE). Le modèle de contexte (Figure 10) précise le contexte d'usage de ces deux sous-tâches de décision en termes de dispositifs. Nous faisons l'hypothèse que le contexte est unique (surface partagée) : ce contexte est décrit par une composition deux objets concrets relatifs aux deux dispositifs d'interaction en entrée.

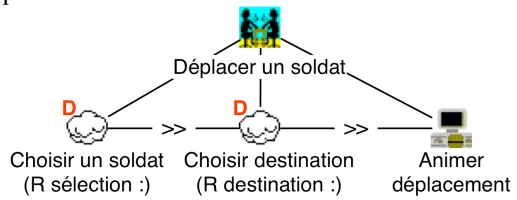


Fig. 8. Tâche [déplacement d'un soldat] avec le système collaboratif et multimodal Warcraft III de la Figure 7 : modèle de tâches coopératives avec la notation Dynamo-Aid.

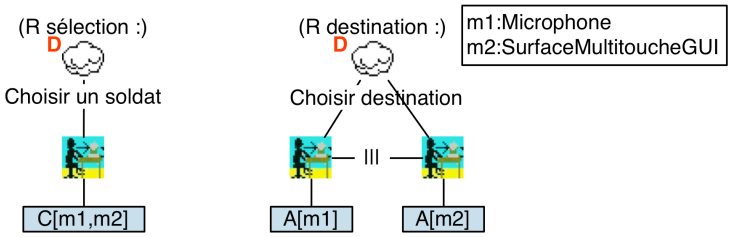


Fig. 9. Tâche [déplacement d'un soldat] avec le système collaboratif et multimodal Warcraft III de la Figure 7 : modèle de tâches individuelles avec la notation Dynamo-Aid.

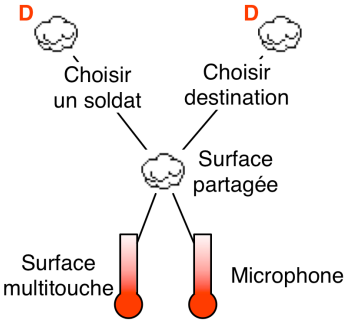


Fig. 10. Tâche [déplacement d'un soldat] avec le système collaboratif et multimodal Warcraft III de la Figure 7 : modèle de contexte avec la notation Dynamo-Aid.

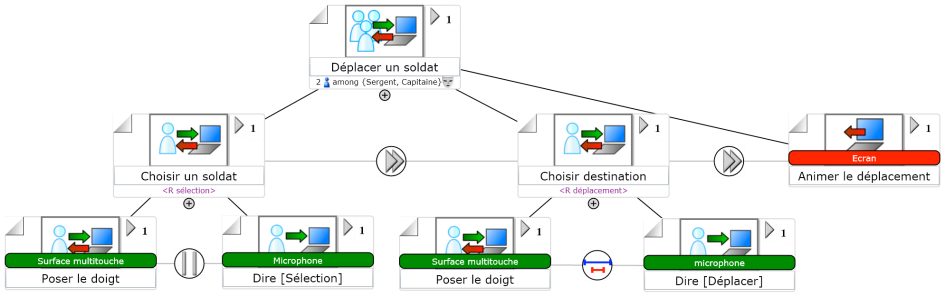


Fig. 11. Tâche [déplacement d'un soldat] avec le système collaboratif et multimodal Warcraft III de la Figure 7 : modèle de tâches avec la notation COMM.

Avec la notation COMM, cette tâche collaborative (« Déplacer un soldat ») est décrite par un arbre unique (Figure 11). Cette tâche collaborative est associée à deux rôles métiers : sergent et capitaine. Elle est également composée de trois sous-tâches dont deux sous-tâches d'interaction individuelle : « Choisir un soldat » et « Choisir destination ». Ces deux sous-tâches d'interaction sont associées respectivement aux deux rôles interactifs suivants : <R sélection> et <R déplacement>. C'est l'usage du rôle interactif qui permet l'expression de la collaboration, ne figeant pas ainsi l'allocation des tâches interactives aux deux rôles métiers. L'interaction multimodale pour accomplir ces deux sous-tâches élémentaires d'interaction est décrite par la combinaison de deux tâches modales, chacune associée à un des deux dispositifs d'interaction en entrée. La forme de l'interaction multimodale est similaire dans les deux cas mais est décrite de deux façons différentes afin mettre en évidence la précision de la description. En effet, dans les deux cas, les modalités sont complémentaires (complémentarité de CARE). Pour la sous-tâche d'interaction « Choisir un soldat », la complémentarité est exprimée par un opérateur de parallélisme. Pour la seconde, la complémentarité est exprimée à l'aide d'un opérateur basé sur les relations de Allen de recouvrement temporel. La seconde description apporte donc plus de précision sur l'ordonnancement temporel pour l'exécution des tâches modales.

Bien que la notation Dynamo-Aid n'est pas originellement prévue pour décrire des tâches à l'aide de la notation CCTT, cet exemple de description montre qu'il est toutefois possible de décrire des tâches multiutilisateurs et multimodales avec Dynamo-Aid étendu avec CCTT. La notation COMM permet de décrire plus finement les tâches multiutilisateur notamment pour exprimer des tâches collaboratives : ce constat n'est pas lié à la notation Dynamo-Aid mais à la notation CCTT.

Du point de vue de la multimodalité, la notation COMM permet une description plus fine de l'articulation temporelle dans l'usage de modalités d'interaction grâce à une extension des opérateurs temporels en considérant les relations de Allen. Par contre, alors qu'avec la notation COMM l'expression des propriétés CARE est implicite car reposant sur des combinaisons de tâches modales à l'aide d'opérateurs, la description est explicite avec la notation Dynamo-Aid. De plus, du fait du modèle de contexte, la notation Dynamo-Aid relie explicitement un contexte d'usage aux tâches, en particulier au regard des dispositifs d'interaction utilisés pour accomplir les tâches concrètes. Cependant, la notation COMM, en plus de la description explicite du dispositif d'interaction pour accomplir une tâche modale, rend explicite les actions atomiques du

langage d'interaction au sein de la description d'une tâche. La notation Dynamo-Aid se limite à la notion de dispositif assimilée à celle de modalité.

5. CONCLUSION

Dans cet article, nous dressons un panorama des notations existantes dédiées à la description des tâches multiutilisateurs et des tâches élémentaires concrètes multimodales, avec une attention particulière portée sur les notations à base de modèles de tâches. Afin d'établir une cartographie, nous identifions et proposons un ensemble de critères couvrant quatre aspects des notations : critères généraux aux notations mais aussi des critères relatifs à l'interaction utilisateur en général, et des critères spécifiques à l'interaction multiutilisateur et multimodale en particulier. Outre un descriptif des notations existantes qui met en évidence leurs caractéristiques principales, nous proposons une synthèse comparative selon notre grille de critères qui souligne les complémentarités ainsi que les différences entre ces notations. Le constat le plus général de cette synthèse comparative est que les notations existantes sont soit générales, soit spécialisées sur l'un des deux aspects, les tâches multiutilisateurs ou les tâches concrètes multimodales. Toutefois, deux notations se démarquent en couvrant explicitement ces deux aspects de l'interaction : les notations COMM et Dynamo-Aid que nous avons illustrées et comparées avec un exemple d'un système multiutilisateur et multimodal.

Pour COMM et Dynamo-Aid les modèles de tâches permettent de décrire les activités multiutilisateurs. La prise en compte de la multimodalité (en particulier les propriétés CARE) établit un lien avec la description du traitement du système qui permet le comportement multimodal décrit au niveau des tâches élémentaires multimodales (ou actions multimodales). Nous soulignons ainsi le lien entre les objectifs des utilisateurs, ses tâches collaboratives, coopératives ou individuelles et les actions mono-modales ou multimodales. Au delà de ce lien entre objectif tâche et action qui est le propre de l'analyse de la tâche, nous identifions avec la notation COMM qu'il ne s'agit pas uniquement d'une juxtaposition de deux notations de description, l'une pour les tâches abstraites et l'autres pour les tâches concrètes. En effet l'étude de la frontière entre tâches abstraites et concrètes (actions multimodales) met en évidence un niveau intermédiaire nouveau, lié au rôle interactif, qui permet l'expression de tâches collaboratives.

REMERCIEMENTS

Le travail présenté dans cet article a été financé par la DGA (Délégation Générale pour l'Armement) sous le contrat PEA FH/PA (Facteurs Humains et Partage d'Autorité).

RÉFÉRENCES

ALLEN, J. F. 1983. Maintaining Knowledge about Temporal Intervals. Communications of the ACM 26, 11, 832-843.

ANNETT, J. AND DUNCAN, K. 1967. Task Analysis and Training Design. Occupational Psychology 41, 211-221.

BALBO, S., OZKAN, N., AND PARIS, C. 2004. Choosing the Right Task modelling Notation: A Taxonomy. Dans The Handbook of Task Analysis for Human-Computer Interaction, Diaper, D. and Stanton N. Editors. Lawrence Erlbaum Associates, Chapter 22, 445-465.

BARON, M., LUCQUIAUD, V., AUTARD, D. AND SCAPIN, D. L. 2006. K-MADe : un Environnement pour le Noyau du Modèle de Description de l'Activité. In Proceedings of the 18th French-speaking ACM Conference on Human-Computer Interaction (IHM'06) (Montréal, Canada, 2006). ACM Press, New York, 287-288.

BASS, L. 1992. A Metamodel for Runtime Architecture of an Interactive System: the UIMS Workshop Tool Developers. SIGCHI Bulletin 24, 1, 32-37.

BOUCHET, J., Nigay, L. and Ganille, T. 2004. ICARE Software Components for Rapidly Developing Multimodal Interfaces. In Proceedings of the international conference on multimodal interfaces (ICMI '04) (State college, USA, October 13-15, 2004). ACM Press, New York, 251-258.

BRUN, P., BEAUDOUIN-LAFON, M., 1995. A Taxonomy and Evaluation of Formalisms for the Specification of Interactive Systems. In Proceedings of the Conference on Human-Computer Interaction, HCI'95, People and Computers X (Huddersfield, UK, 1995). University Press, 197-212.

CARR, D. A. 1994. Specification of interface interaction objects. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'94) (Boston, USA, April 24-28, 1994). ACM Press, New-York, 372-378.

CLERCKX, T., VANDERVELPEN, C. AND CONINX, K. 2007. Task-Based Design and Runtime Support for Multimodal User Interface Distribution. In Proceedings of the Engineering Interactive Systems Conference (EIS'07) (Salamanca, Spain, March 22-24, 2007). Springer, Heidelberg, 89-105.

COUTAZ, J., NIGAY, L., SALBER, D., BLANDFORD, A., MAY, J. AND YOUNG, R. M. 1995. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE Properties. In Proceedings of the IFIP International Conference on Human Computer Interaction (INTERACT'95) (Lillehammer, Norway, June 25-29, 1995). Chapman & Hall, 115-120.

DIX, A., FINLAY, J., ABOWD, D.G. AND BEALE, R. 2004. Human computer interaction. Pearson Prentice Hall, 834 pages.

DRAGICEVIC , P., FEKETE, J. D. 2004. Support for input adaptability in the icon toolkit. In Proceedings of the international conference on multimodal interfaces (ICMI '04) (State college, USA, October 13-15, 2004). ACM Press, New York, 212-219.

HARTSON, H. R., SIOCHI, A. C. AND HIX, D. 1990. The UAN: a User-Oriented Representation for Direct Manipulation Interface Designs. Transactions on Information Systems 8, 3, 181-203.

ISO 8807. 1989. LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.

JOURDE, F. 2011. Collecticiel et Multimodalité : spécification de l'interaction, la notation COMM et l'éditeur e-COMM. Thèse de doctorat de l'Université de Grenoble, 301 pages.

JOURDE, F., LAURILLAU, Y. AND NIGAY, L. 2010. COMM Notation for Specifying Collaborative and MultiModal Interactive Systems. In Proceedings of the ACM International Conference on Engineering Interactive Computing Systems (EICS'10) (Berlin, Germany, 19-23 June, 2010). ACM Press, New York, 125-134.

JOURDE, F., LAURILLAU, Y. AND NIGAY, L. 2010. e-COMM, un éditeur pour spécifier l'interaction multimodale et multiutilisateur. In Proceedings of the 22nd French-speaking ACM Conference on Human-Computer Interaction (IHM'10) (Luxembourg, Luxembourg, September 20-23, 2010). ACM Press, New-York, 225-228.

JOURDE, F., LAURILLAU, Y., MORAN, A., AND NIGAY, L. 2008. Towards Specifying Multimodal Collaborative User Interfaces: A Comparison of Collaboration Notations. In Proceedings of the Design, Specification and Verification of Interactive Systems Conference (DSV-IS'08) (Kingston, Canada, July 16-18, 2008). Lecture Notes in Computer Science 5136, Springer, Heidelberg, 281-286.

LALANNE, D. NIGAY, L., PALANQUE, P., ROBINSON, P., VANDERDONCKT, J. AND LADRY, J-F. 2009. Fusion Engines for Input Multimodal Interfaces: a Surveys. In Proceedings of the International Conference on Multimodal Interfaces (ICMI'09) (Cambridge, USA, November 2-4, 2009). ACM Press, New-York, 2009, 153-160.

LIM, Y. K. 2004. Multiple Aspect Based Task Analysis (MABTA) for User Requirements Gathering in Hightly-contextualized Interactive System Design. In Proceedings of the 3rd annual conference on Task models and diagrams (TAMODIA'04) (Prague, Czech Republic, November 15-16, 2004). ACM Press, New York, 7-15.

LIMBOURG, Q., VANDERDONCKT, J. 2004. Comparing Task Models for User Interface Design. In The Handbook of Task Analysis for Human-Computer Interaction, Diaper, D. and Stanton N. Editors. Lawrence Erlbaum Associates, Chapter 6, 135-154.

LUCQUIAUD, V. 2005. Proposition d'un Noyau et d'une Structure pour les Modèles de Tâches Orientés Utilisateurs. In Proceedings of the 17th French-speaking ACM Conference on Human-Computer Interaction (IHM'05) (Toulouse, France, September 27-30, 2005). ACM Press, New-York, 83-91.

MARTINIE DE ALMEIDA, C. 2011. Une approche à base de modèles synergiques pour la prise en compte simultanée de l'utilisabilité, la fiabilité et l'opérabilité des systèmes interactifs critiques. Thèse de doctorat de l'Université de Toulouse, 254 pages.

MARTINIE, C., BARBONI, E., NAVARRE, D., PALANQUE, P., FAHSSI, R., POUPART, E., CUBERO-CASTAN, E. 2014. Multi-models-based engineering of collaborative systems: application to collision avoidance operations for spacecraft. In Proceedings of the ACM International Conference on Engineering Interactive Computing Systems (EICS'14) (Rome, Italy, 17-20 June, 2014). ACM Press, New York, 85–94.

MEJIA, D.A., MORAN, A.L. AND FAVELA, J. 2007. Supporting Informal Co-located Collaboration in Hospital Work. In Proceedings of the Collaboration Researchers International Working Group Workshop (CRIWG'07) (Bariloche, Argentina, September 16-20, 2007). Lecture Notes in Computer Science 4715, Springer, Heidelberg, 255-270.

MOLINA, A.I., REDONDO, M.A., ORTEGA, M. AND HOPPE, U. 2008. CIAM: A Methodology for the Development of Groupware User Interfaces. Universal Computer Science14, 9, 1435-1446.

MOLINA, A.I., GIRALDO, W.J., REDONDO, M.A. AND ORTEGA, M. 2007. A Proposal of Integration of the GUI Development of Groupware Applications into the Software Development Process. In Proceedings of the Collaboration Researchers International Working Group Workshop (CRIWG'07) (Bariloche, Argentina, September 16-20, 2007). Lecture Notes in Computer Science 4715, Springer, Heidelberg, 111-126.

MOLINA, A. I., REDONDO, M. A. AND ORTEGA, M. 2009. A Review of Notations for Conceptual Modeling of Groupware Systems. In New Trends on Human-Computer Interaction, Macías, J., Granollers Saltiveri, A., Latorre, P. Editors. Springer, 75-86.

MORI, G., PATERNÓ, F. AND SANTORO, C. 2002. CTTE : Support for Developing and Analyzing Task Models for Interactive System Design. Transactions on software engineering 28, 8, 797-813.

NAVARRE, D., PALANQUE, P., LADRY J.F. AND BARNONI, E. 2009. ICOs. Transactions on Computer-Human Interaction 16, 4, 1-56.

NIGAY, L. AND COUTAZ, J. 1995. A Generic Platform for Addressing the Multimodal Challenge. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'95) (Denver, USA, May 7-11, 1995). ACM Press, New-York, 98- 105.

PATERNO, F., SANTORO, C. AND TAHMASSEBI, S. 1998. Formal Models for Cooperative Tasks: Concepts and an Application for En-Route Air-Traffic Control. In Proceedings of the Design, Specification and Verification of Interactive Systems Conference (DSV-IS'98) (Abingdon, UK, June 3-5, 1998). Springer-Verlag, Wien, 71–86.

PATERNÓ, F. 1999. Model-Based Design and Evaluation of Interactive Application. Springer-Verlag London, UK.

PENICHET, V. M. R., LOZANO, M. D., GALLUD, J. A. AND TESORIERO, R. 2010. Requirement-based Approach for Groupware Environments Design. Systems and Software 83, 8, 1478-1488.

ROOK, P. 1986. Controlling software projects. Software Engineering 1, 1, 7-16.

ROSCHELLE, J. AND TEASLEY, S.D. 1995. The Construction of Shared Knowledge in Collaborative Problem Solving. Computer Supported Collaborative Learning, 69–97.

SAFIN, S. AND LECLERCQ, P. 2009. User Studies of a Sketch-Based Collaborative Distant Design Solution in Industrial Context. In Proceedings of the 6th international Conference on Cooperative Design, Visualization and Engineering (CDVE'09) (Luxembourg, Luxembourg, September 20-23, 2009). Lecture Notes in Computer Science 5738, Springer, Heidelberg, 117-124.

SCAPIN, D. L. AND PIERRET-GOLBREICH, C. 1990. Towards a method for task description: MAD. In Proceedings of the conference on Work with Display Units WWDU 89 (Montréal, Canada, September 11-14, 1989). Elsevier, North-Holland, 371-380.

SERRANO, M. AND NIGAY, L. 2006. Multimodal Interaction on Mobile Phones: Development and Evaluation Using ACICARE. In Proceedings of the 8th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI'06) (Epoo, Finland, September 12-15, 2006). ACM Press, New-York, 129-136

SERRANO, M. AND NIGAY, L. 2009. Temporal aspects of CARE-based multimodal fusion: from a fusion mechanism to composition components and WoZ components. In Proceedings of the International Conference on Multimodal Interfaces (ICMI'09), (Cambridge, USA, November 2-4, 2009). ACM Press, New-York, 177-184.

SERRANO, M., NIGAY, L, LAWSON, J-Y, RAMSAY, A, MURRAY-SMITH, R. AND DENEF, S. 2008. The OpenInterface framework: a tool for multimodal interaction. Extented Abstracts on Human Factors in Computing Systems (CHI EA'08) (Firenze, Italy, April 5-10, 2008). ACM Press, New-York, 3501-3506.

SINNIG, D., WURDEL, M., FORBRIG, P., CHALIN, P. AND KHENDEK, F. 2007. Practical Extensions for Task Models. In Proceedings of the 6th International Workshop on Task Models and Diagrams for User Interface Design (TAMODIA'07) (Toulouse, France, November 7-9, 2007). Lecture Notes in Computer Science 4849, Springer, Heidelberg, 42-55.

TARBY, J. C. AND BARTHET, M. F. 2001. Analyse et Modélisation des Tâches dans la Conception des Systèmes d'Information : la Méthode Diane+. Dans Analyse et conception de l'IHM, Interaction pour les systèmes d'information, volume 1. Hermes, Paris, Chapitre 4, 117-144.

TSE, E., GREENBERG, S., SHEN, S., FORLINES, C. AND KODOMA, R. 2008. Exploring True Multi-User Multimodal Interaction over a Digital Table. In Proceedings of the 7th ACM Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS'08) (Cape Town, South Africa, February 25-27, 2008). ACM Press, New York, 109-118.

VAN DER VEER, G., VAN WELIE, M. AND CHISALITA, C. 2002. Introduction to Groupware Task Analysis. In Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design (TAMODIA'02) (Bucarest, Romania, July 18-19, 2002). INFOREC Publishing House, Bucharest, 32-39.

VAN DER VEER, G. AND VAN WELIE, M. 2000. Task Based Groupware Design: Putting Theory into Practice. In Proceedings of the 3rd ACM Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS'00) (New-York, USA, August 17-19, 2000). ACM Press, New-York, 326-337.

VAN WELIE, M., VAN DER VEER, G. C. AND ELIËNS, A. 1998. An ontology for task world models. Design, Specification and Verification of Interactive System. In Proceedings of the Design, Specification and Verification of Interactive Systems Conference (DSV-IS'98) (Abingdon, UK, June 3-5, 1998). Springer-Verlag, Wien, 57-70.

VANACKEN, D., DE BOECK, J., RAYMAEKERS, C., AND CONINX, K. 2006. NiMMiT: A notation for modeling multimodal interaction techniques. In Proceedings of the First International Conference on Computer Graphics Theory and Applications (GRAPP'06) (Setúbal, Portugal, February 25-28, 2006). INSTICC, 224–231.

VENEMA, D. C. 1999. The N-NUAN; a New User Action Notation, Thesis, Faculty of Sciences, division Mathematics and Computer Science, University of Amsterdam, Netherlands, 43 pages.

VERNIER, F. AND NIGAY, L. 2000. A Framework for the Combination and Characterization of Output Modalities. In Proceedings of the 7th International Workshop in Interactive Systems: Design, Specification, and Verification (DSV-IS'00) (Limerick, Ireland, June 5-6, 2000). Springer-Verlag, Berlin Heidelberg, 32-48.

VILLAREN, T. 2012a. A Modèles et mécanismes d'adaptation de l'interaction homme-machine aux changements de contexte. Thèse de doctorat, Télécom Bretagne, 335 pages.

VILLAREN, T., LEAL, A., AND COPPIN, G. 2012b. Contribution de la méthodologie SAMANTA à l'ergonomie d'interfaces en support des changements de tâche. In Actes de la conférence Ergo'IHM 2012, Biarritz, France.

WURDEL, M., SINNIG, D. AND FORBRIG, P. 2008. CTML: Domain and Task Modeling for Collaborative Environments. Universal Computer Science 14, 19, 3188-3201.

Frédéric Jourde est Docteur en informatique. Il a effectué ses travaux de thèse au laboratoire LIG au sein de l'équipe IIHM, dans le domaine de l'Interaction Homme-Machine, et plus précisément sur l'interaction multimodale et multiutilisateur. Ses contributions, et en particulier la notation COMM et l'éditeur e-COMM, ont été utilisés dans le cadre d'un projet industriel d'envergure dédié au poste de commande de drones militaires et depuis cinq ans pour l'enseignement des collecticiels aux étudiants du Master 2 Professionnel Génie Informatique.

Yann Laurillau est Maître de Conférences à l'Université Grenoble-Alpes et effectue sa recherche au sein de l'équipe IIHM du laboratoire LIG. Le thème central de sa recherche est l'Interaction Homme-Machine et, plus particulièrement, la conception et le développement de systèmes interactifs multiutilisateur ou collecticiels. C'est un thème développé depuis plus de 10 ans. Les résultats sont à la fois conceptuels et techniques par le biais de réalisations logicielles. Les surfaces interactives ainsi que l'interaction gestuelle tangible sont également abordés dans ses travaux de recherche.

Laurence Nigay est Professeur à l'Université Grenoble-Alpes et dirige l'équipe IIHM (Ingénierie de l'Interaction Homme-Machine) du laboratoire LIG. L'équipe IIHM comprend 11 membres permanents et plus de 20 membres non-permanents, doctorants, post-doctorants, visiteurs scientifiques et ingénieurs. Les travaux de recherche de L. Nigay s'inscrivent dans le domaine de l'Interaction Homme-Machine. Ses contributions concernant l'interaction multimodale sont d'abord conceptuelles comme la définition d'une modalité et de relations entre modalités ainsi que l'établissement d'espaces de conception. Ses contributions concernent aussi l'ingénierie de l'interaction multimodale avec la définition de moteurs de fusion et de plusieurs plateformes logicielles pour l'interaction multimodale : ICARE, OpenInterface (projet européen FP6 coordonné par L. Nigay) et DynaMO.

Un modèle de tâches exploitable à l'exécution pour une assistance à l'utilisateur dans les environnements ambiants

Asma Gharsellaoui, Yacine Bellik, Christophe Jacquet

Un modèle de tâches exploitable à l'exécution pour une assistance à l'utilisateur dans les environnements ambiants

ASMA GHARSELLAOUI
YACINE BELLIK
LIMSI-CNRS – Université Paris-Sud
CHRISTOPHE JACQUET
Supélec – Gif-sur-Yvette

Résumé : Les modèles de tâches existants ont souvent été utilisés dans le cadre des systèmes interactifs graphiques. Dans cet article, nous proposons d'utiliser le modèle de tâches dans un environnement ambiant au moment de l'exécution des tâches par l'utilisateur, afin de suivre les actions de l'utilisateur, vérifier qu'il n'a pas fait d'erreurs lors de l'accomplissement de ses tâches et lui procurer de l'aide quand cela est nécessaire. En particulier, nous présentons, dans une première contribution, un modèle de tâches spécifique aux environnements ambiants. Ce modèle permet d'associer des caractéristiques dynamiques à chaque tâche permettant ainsi à un système de supervision d'attribuer des états aux tâches au moment de l'exécution en fonction des informations échangées avec l'environnement (démarrage d'une tâche, fin de réalisation d'une tâche, états des pré-conditions...). Notre deuxième contribution consiste en un système de suivi et d'assistance qui exploite notre modèle de tâches. Plus précisément nous spécifions la stratégie d'intervention du système qui va orienter l'utilisateur. Nous présentons par la suite une illustration de notre système à travers le déroulement d'un scénario sur notre simulateur. Cette simulation montre comment les interactions avec le modèle de tâches à l'exécution nous permettent de produire un système dynamique, qui prend en considération le contexte et fournit une aide à l'utilisateur pour la réalisation de ses tâches quotidiennes. Enfin, nous terminons par une conclusion sur notre approche et les perspectives ouvertes pour ce travail.

Mots clés : Modélisation des tâches utilisateurs, intelligence ambiante, interactions dans les environnements ambiants, systèmes d'aide et de suivi.

Abstract: Existing task models have often been used in the context of graphic systems. In this paper, we propose to use the task model at runtime to monitor user actions, to verify that he/she has not made a mistake when performing his/her actions and to give him/her help when necessary. In particular, we present, as a first contribution, a task model specific to interactions in ambient environments. This model enables to assign dynamic characteristics to each task thereby allowing to a supervision system to assign states to tasks at runtime based on the information exchanged with the environment (start of a task, end of a task, pre-conditions states...). Our second contribution is a monitoring and support system that exploits our task model. More precisely we specify the intervention strategy of our system in order to guide the user. We present then an illustration of our system through the execution of a scenario on our simulator. This simulation shows how the interactions with the task model at runtime allow us to produce a dynamic system that takes into consideration the context and provides assistance to the users while carrying out their daily tasks. Finally, we end with a conclusion and perspectives of our approach.

Key words: User Task modeling, Ambient intelligent environments, Ambient Interactions, Monitoring and assistance system.

Adresse des auteurs : Asma Gharsellaoui (asma.gharsellaoui@limsi.fr), LIMSI, Bât 508, Plateau du Moulon, B.P. 133, 91403, Orsay Cedex France. Yacine Bellik (yacine.Bellik@limsi.fr), LIMSI, Bât 508, Plateau du Moulon, B.P. 133, 91403, Orsay Cedex France. Christophe Jacquet (Christophe.Jacquet@supelec.fr), Supélec- Département informatique,3 rue Joliot-Curie, 91192 Gif-sur-Yvette Cedex,France.

1. INTRODUCTION

Le terme intelligence ambiante [1] a été introduit initialement par la Commission Européenne en 2001 [Ducatel et al. 2001], [Riva et al. 2003], bien que le concept en lui-même remonte au début des années 90 [Weiser 1991] [Weiser 1993]. Le concept d'environnement ambiant (intelligence ambiante) réfère à un environnement quotidien pour l'utilisateur mais dans lequel la technologie est omniprésente et discrète [Riva 2005]. Dans ce cadre plusieurs domaines d'applications sont concernés [Friedewald and Costa 2003] comme par exemple les maisons intelligentes [Gerhart 1999] [Harper 2003] [Punie 2003]. Les systèmes d'intelligence ambiante ont donc pour objectif de non seulement percevoir l'environnement physique mais également d'agir sur celui-ci de façon à s'intégrer dans les activités des utilisateurs de la façon la plus harmonieuse possible pour les assister en cas de besoin.

Pour atteindre cet objectif, le système doit disposer d'un modèle de référence décrivant les tâches que l'utilisateur doit effectuer : le modèle de tâches. [Barthet 1988] a introduit la notion de « tâche prévue » telle qu'imaginée par le concepteur et la notion de « tâche effective » qui retrace le plan de l'activité observable de l'utilisateur. La comparaison de ces deux éléments s'appuie sur le modèle de tâches. En effet un modèle de tâches décrit les actions destinées à être effectuées afin d'atteindre les objectifs de l'utilisateur [Card et al. 1983] [Schulungbaum 1996] et les différentes manières de les atteindre [Ormerod and Shepherd 2004]. La modélisation des tâches vise en particulier à construire un modèle qui décrit précisément les relations entre les différentes tâches [Paterno 2002]. L'utilisation des modèles de tâches est intéressante dans le cadre des environnements ambiants car ceux-ci disposent de capteurs qui peuvent renseigner le système sur les états des tâches à l'exécution.

Plusieurs travaux se sont déjà intéressés à l'assistance de l'utilisateur final dans les environnements ambiants. A titre d'exemple, dans [Sassi et al. 2010], les auteurs proposent un système à base d'agents virtuels qui interagissent avec les utilisateurs afin de les assister. Ce système dispose de certaines informations sur l'utilisateur et sur son environnement et en collecte d'autres en interagissant avec lui (l'utilisateur). Il effectue ensuite des raisonnements lui permettant de : (1) déterminer les besoins potentiels des utilisateurs (en terme de services systèmes) et démarrer les services en question ou (2) répondre à une sollicitation de la part de l'utilisateur (exploitation d'un service donné). Nous avons constaté que outre ces deux interventions du système, l'utilisateur aura besoin de suivi au cours de la réalisation de ses tâches et d'assistance en cas de dysfonctionnement. Cet aspect reste non entièrement couvert par les travaux existants.

Notre objectif est de proposer un système de supervision et d'assistance qui possédera un modèle des tâches (permettant de décrire ce qui devrait être fait), et qui devra être capable de situer la tâche en cours dans ce modèle. À partir des données issues de capteurs sur l'état de l'environnement (les objets manipulés, la position de l'utilisateur, la tâche en cours de réalisation...), ce système proactif devrait être capable de suivre le déroulement des tâches utilisateur et de décider d'intervenir lui même à un instant donné pour assister l'utilisateur en cas de besoin. Notre travail se focalise donc sur la définition de méthodes garantissant le bon déroulement des tâches utilisateur (c'est à dire conforme à ce qui est prévu dans le modèle) en mettant en œuvre une approche de modélisation des tâches adaptée aux caractéristiques des environnements ambiants.

1. Ambient Intelligence (AmI)

Outre l'identification des actions à accomplir dans le cadre de l'activité de l'utilisateur, nous souhaitons automatiser un algorithme de suivi et donc nous nous basons sur des caractéristiques quantifiables telles que la durée. Dans ce but nous nous proposons de prendre en compte de façon précise le facteur temporel dans la réalisations de ces tâches. Pour cela nous avons besoin de réaliser un simulateur de modèle de tâches qui nécessite :

– l'ajout de certaines caractéristiques dans un formalisme de modèle de tâches
– la définition d'un algorithme d'exécution (que nous avons choisi de basé sur la durée des tâches)
– le « branchement » des tâches élémentaires avec l'environnement ambiant

Ce papier présente les deux premières étapes de la réalisation de l'outil. Le résultat est illustré sur un scénario de cuisine (voir section 6). Dans la deuxième section de cet article, nous nous intéressons aux spécificités de la modélisation des tâches réalisées dans un environnement ambiant. Dans la troisième section, nous nous intéressons aux modèles de tâches exploitables au moment de l'exécution des tâches par l'utilisateur. La quatrième section présente les détails du modèle de tâches que nous proposons, et qui est exploitable pendant l'exécution des tâches réalisées dans un environnement ambiant. Le système d'aide et de suivi est ensuite présenté en détaillant ses stratégies d'intervention. Dans la dernière partie, nous présentons un outil de simulation des stratégies d'intervention et nous illustrons notre démarche au travers son utilisation sur un cas d'usage.

2. SPÉCIFICITÉS DE LA MODÉLISATION DES TÂCHES RÉALISÉES DANS UN ENVIRONNEMENT AMBIANT

Dans un travail précédent [Gharsellaoui et al. 2012], nous avons mené une étude pour identifier les besoins de la modélisation des tâches réalisées dans un environnement ambiant. La spécificité des environnements ambiants induit des conséquences aussi bien sur le modèle de tâches en lui-même que sur les algorithmes de suivi. Ainsi le modèle doit être enrichi pour prendre en compte des spécificités qui n'existent pas dans les modèles classiques. Par exemple certaines tâches ne peuvent avoir lieu que dans un endroit particulier (Par exemple, la tâche « préparer à manger » ne peut s'effectuer que dans la cuisine). Une autre nécessitera des ressources physiques et/ou logicielles (services) qui peuvent être disponibles ou pas à un instant donné. Ces informations ont également un impact sur l'assistance produite puisqu'elles vont permettre de fournir une meilleure explication à l'utilisateur sur les raisons qui empêchent une tâche donnée d'être réalisable. Il nous est alors apparu que le modèle des tâches réalisées dans un environnement ambiant doit inclure la possibilité d'étiqueter une tâche selon des contraintes spatiales et permettre de spécifier les ressources nécessaires à sa réalisation. Nous avons également abordé dans cette étude le problème de la détermination du niveau de granularité de la tâche à modéliser. Nous avons ainsi proposé d'arrêter la décomposition au niveau où les services du système sont invoqués à l'exception des tâches purement utilisateur (qui ne reposent pas sur des services logiciels).

Une autre exigence pour les modèles de tâches utilisés dans le cadre des environnements ambiants (sans être exclusive à ce domaine) est que le modèle doit être exploitable pendant l'exécution des tâches par l'utilisateur afin de pouvoir lui procurer une aide au moment opportun. La section suivante sera consacrée à ce type de modèle.

3. ÉTUDE DES MODÈLES DE TÂCHES EXISTANTS : UTILISATION AU MOMENT DE L'EXÉCUTION ET POUVOIR D'EXPRESSION

Notre but étant d'assister l'utilisateur au moment même de la réalisation de ses tâches, le modèle sur lequel nous nous reposons doit être exploitable en temps réel. Dans ce qui suit nous passons en revue quelques exemples d'utilisation de modèles de tâches au moment de l'exécution.

3.1 Exemples d'utilisation de modèles de tâches à l'exécution

Un grand nombre de modèles de tâches a été développé, en particulier dans le contexte des interfaces graphiques : HTA [Annett and Duncan 1967], GOMS [Card et al. 1983] [John and Kieras 1996], TKS [Johnson et al. 1984] [Johnson 1992], UAN [Siochi and Hartson 1989], MAD [Scapin and Pierret-Golbreich 1989], GTA [VanderVeer et al. 1996], DIANE+ [Tarby and Barthet 1996], CTT [Paterno 1999], TOOD [Mahfoudhi et al. 2001], K-MAD [Baron et al. 2006] et HAMSTERS [Martinie De Almeida et al. 2011]. Une étude comparative dans [Limbourg and Vanderdonckt 2003] a montré que certaines propriétés doivent être prises en charge dans les modèles de tâches comme par exemple : la description des buts, une structure hiérarchique pour la modélisation des tâches, l'intégration d'opérateurs temporels décrivant les relations temporelles entre tâches et le détail des objets et des actions qui permettront de détailler la modélisation des interfaces utilisateurs. Cette étude montre que chaque type de modèle de tâches peut être utile pour un type d'application donné. Les auteurs ont comparé les modèles de tâches existants selon deux axes : leur pouvoir d'expression et leur complexité. Cette étude a montré que plus les modèles de tâches sont expressifs plus ils deviennent complexes.

Certains travaux se sont déjà intéressés à l'utilisation du modèle de tâches au moment de l'exécution d'une IHM. Plus précisément, dans le but d'assister l'utilisateur lors de son interaction, [Petoud 1990] a utilisé des graphes d'enchaînements afin de représenter l'évolution du dialogue au cours d'une session. Ce graphe représente les opérations proposées par le système et les liens d'ordonnancement entre elles. Ce graphe a été utilisé conjointement avec un tableau de bord qui permettait de renseigner l'utilisateur sur l'état de l'opération en cours [Tarby 1993].

Dans le domaine de la génération automatique d'interfaces utilisateur, Klug et Kangasharju [Klug and Kangasharju 2005] présentent un modèle de tâches exécutable exploité pour générer une interface graphique dynamique composée de blocs d'interface préprogrammés. Ils étendent la notation de CTT (ConcurTaskTree) en donnant quatre états dynamiques aux tâches feuilles (Inactive, Permise, Suspendue, Active) permettant ainsi l'exploitation du modèle de tâches à l'exécution. En outre, les auteurs proposent de rajouter des ports d'entrée et de sortie aux opérateurs temporels afin de faciliter l'échange d'information entre les tâches.

Une autre utilisation des modèles de tâches au moment de l'exécution a été présenté dans [Sottet et al. 2008] dans le cadre de la plasticité des interfaces. Dans ce cadre, le modèle de tâches est utilisé conjointement avec le modèle conceptuel, le modèle de l'espace de travail et le modèle d'interaction afin de permettre l'adaptation des systèmes interactifs au contexte d'utilisation. Ces différents modèles constituent les contraintes auxquelles doit obéir le système dynamiquement et ce en définissant des règles de transformation de ces modèles en composants logiciels répondant aux changements dans le contexte d'utilisation. Une interface utilisateur (UI) est alors définie comme un graphe de modèles décrivant l'in-

terface utilisateur à partir de différents points. Des mappages sont définis pour décrire les fonctions de transformations dynamiques à l'exécution les plus appropriées aux nouveaux contextes d'utilisation.

Dans ce même contexte d'adaptation d'IHM en temps réel, le framework UsiComp [Frey et al. 2012] a été proposé. Usicomp permet au concepteur de créer et de modifier les modèles au moment de la conception aussi bien qu'en temps réel. Il offre deux modules : le premier module est dédié à la conception des interfaces utilisateur et se base sur la notation CTT alors que le deuxième module (d'exécution) est responsable de la génération automatique des interfaces utilisateur selon un ensemble de règles de transformation.

Dans [Blumendorf et al. 2010], les auteurs proposent également d'utiliser le modèle de tâches pour la création d'interfaces utilisateurs adaptatives. La mise à jour du modèle de tâches pendant l'exécution permet d'avoir une idée sur le contexte afin de produire l'interface la mieux adaptée [Roscher et al. 2011]. Des caractéristiques dynamiques ont été ajoutées au modèle permettant ainsi sa mise à jour en fonction des événements qui ont lieu. Le modèle CTT a été réutilisé dans ce travail et adapté : exploitation de quelques états des tâches (active, possible et réalisée), attribution des informations relatives au contexte (la position) et différentiation entre deux types de tâches (les tâches interactives en entrée ou en sortie).

Toujours dans le contexte de la conception et du développement d'interfaces utilisateur, dans [Stephanidis et al. 1998] les auteurs proposent d'adapter en temps réel les interfaces graphiques en fonction des capacités, des connaissances, des exigences et des préférences de l'utilisateur. Pour ce faire, ils proposent la notion de tâche polymorphe. Les différentes tâches sont structurées d'une manière hiérarchique et décomposées progressivement de façon polymorphe définissant ainsi les différentes alternatives pour la réalisation de chaque tâche. Suite à une session d'initiation, l'interface est adaptée aux caractéristiques de l'utilisateur et est modifiée dynamiquement au moment de l'exécution en fonction des interactions avec ce dernier (par exemple en fonction du taux d'erreurs ou encore d'échec dans la réalisation d'une tâche donnée...).

Les modèles de tâches exploitables à l'exécution ont également été utilisés, dans [Martinie et al. 2014], pour la distribution dynamique des interfaces utilisateurs dans le cadre des systèmes interactifs critiques. Ici, le modèle HAMSTERS [Martinie De Almeida et al. 2011] a été utilisé pour décrire les opérations possibles lors de scénarios "catastrophe". Toutes évolution dans le contexte affecte le modèle de tâches. Ce dernier est utilisé conjointement avec le modèle de système et toutes modification dans l'un des modèles induit une modification dans l'autre pour garantir la cohérence entre les contenus des deux modèles. Un processus de vérification de cohérence entre ces deux modèles et l'interface générée est effectué en permanence.

D'autres travaux ont eu recours à l'usage des modèles de tâches pour la génération de composants orientés tâches et un outil a été proposé à cet effet dans [Bourguin et al. 2007]. Le modèle de tâches décrit le comportement attendu du composant et sert de point de départ pour la génération des composants orientés tâche. Ce modèle correspond à une fusion des modèles CTT et K-MAD. Ceci se manifeste par : la vue arborescente de CTTE, les icônes de K-MADe et la fusion des opérateurs temporels de ces deux modèles. Ce modèle est simple et plusieurs fonctionnalités ont été allégées. Ceci s'explique par le fait que le but principal de ce modèle est de définir les composants à partir des descriptions qui y sont contenues. La focalisation majeure était plutôt sur les méthodes clés qui seront

implémentées dans le composant en question.

3.2 Étude du pouvoir d'expression de quelques modèles de tâches exploitables à l'exécution

Parmi les travaux de la littérature qui peuvent présenter un apport intéressant par rapport à notre problématique nous citerons ProtoTask [Lachaume et al. 2012] et HAMSTERS [Martinie De Almeida et al. 2011]. ProtoTask est un simulateur qui a été développé pour simuler l'exécution d'un modèle de tâches K-MAD. Ce simulateur exploite des pré-conditions et les évalue pour déterminer les tâches pouvant être démarrées parmi l'ensemble des tâches faisables à l'étape courante (les boutons correspondants sur l'interface du simulateur sont grisés ou cliquables). K-MAD permet d'exprimer des durées d'exécution mais de manière informelle, c'est pourquoi ProtoTask qui simule l'exécution d'un modèle K-MAD ne prend pas en compte ces durées . HAMSTERS, quant à lui, a été introduit en tant que support à la conception et au développement de systèmes interactifs. Il repose sur l'utilisation des opérateurs temporels de CTT et propose un affinement des types de tâches (en ajoutant des tâches utilisateurs cognitive, perceptive et motrice ainsi que des tâches interactives d'entrée, de sortie et d'entrée sortie). Ce modèle définit uniquement trois états pour les tâches : effectuée, disponible et désactivée. Par ailleurs HAMSTERS vérifie les contraintes temporelles mais dans le but de prévoir des actions système qui sont en adéquation avec les durées des tâches utilisateurs.

Dans notre étude, nous nous intéressons aussi à la notation COMM [Jourde et al. 2009] qui a été introduite pour la description de l'interaction multimodale et collaborative. Elle propose un affinement des types de tâches (Tâche de calcul, de présentation, mentale, d'action, d'interaction, de coordination, d'action de groupe, d'interaction de groupe). Ce modèle reprend les opérateurs temporels de CTT et les opérateurs temporels d'ALLEN [Allen 1983]. Ainsi à titre d'exemple l'opérateur exprimant le parallélisme est affiné en trois opérateurs exprimant : la coincidence, la concomittance et le parallélisme et celui exprimant le séquencement est affiné en deux pour exprimer : l'anachronisme et la séquence. Cette notation dispose d'une application web d'édition et de simulation. Les pré-conditions et les post-conditions proposés dans cet outils constituent un champs texte qui n'est pas calculable.

Nous pouvons également considérer CTML [Wurdel et al. 2009] qui est un langage de modélisation développé comme extension du modèle de tâches CTT. Ce langage de modélisation de tâches coopératives a été développé dans le but de permettre d'exprimer des scénarios dans les environnements ambiants. CTML intègre un modèle de coopération qui permet de modéliser les tâches coopératives et repose sur un comportement basé sur les rôles en décrivant leurs pré-conditions et effets. Il permet de prendre en considération la dépendance de la localisation des objets et la modélisation des périphériques invoqués mais il ne permet pas d'exprimer des contraintes temporelles sur les tâches (durées...) ni de spécifier des états dynamiques à l'exécution des tâches.

La table I présente une synthèse comparative de différents modèles et outils de modélisation des tâches étudiés selon six critères :

- Opérateurs temporels : indique les opérateurs temporels supportés par le modèle.
- États dynamiques des tâches : indique les différents états possibles d'une tâche.
- Spécification des conditions : indique le pouvoir d'expression des préconditions permettant de conditionner la réalisation des tâches.
- Spécification des performances temporelles : indique les informations prises en

compte par le modèle concernant la gestion temporelle des tâches.
– Outils : indique les types d'outils logiciels accompagnant le modèle.
– Types de tâches : indique les types de tâches supportés par le modèle.

3.3 Discussion

Nous avons constaté que les modèles existants gèrent bien les contraintes temporelles inter-tâches (opérateurs temporels). Bien que ces modèles permettent d'exprimer les contraintes temporelles intra-tâches (durée minimale ou maximale), celles-ci ne sont pas exploitées au moment de l'exécution. Par ailleurs, CTML mis à part, aucun des modèles de tâches existants n'offre la possibilité d'affecter à une tâche un dispositif particulier ou un endroit dans lequel elle devrait avoir lieu. Malheureusement, l'éditeur et le simulateur de CTML ne sont pas disponibles publiquement, ce qui empêche sa réutilisation.

Par ailleurs, il est utile de disposer d'un simulateur pour dérouler les tâches sur un scénario donné. Cependant, les simulateurs de K-MADe (Prototask y compris) et de CTTE ne tiennent pas compte de l'état de la tâche à l'exécution. En effet, les simulateurs calculent un ensemble de tâches pouvant être réalisées à la prochaine étape (appelé Enabled Task Set dans CTT). Une fois que l'utilisateur sélectionne une tâche donnée pour indiquer qu'elle est en train d'être exécutée, celle-ci est automatiquement réalisée du début à la fin. Les tâches élémentaires sont considérées comme étant atomiques à l'exécution. Il est donc impossible pour deux tâches feuilles d'être actives en même temps, ce qui contredit la spécification de la simultanéité définie dans le modèle. Or comme nous l'avons vu précédemment, nous avons besoin d'un modèle qui puisse être mis à jour au moment de l'exécution (états des tâches, durée...) afin de pouvoir suivre le déroulement des tâches au moment où elles sont accomplies. Pour représenter correctement les tâches du monde réel, l'utilisation des états actifs doit être affectée aux tâches feuilles. Les états des tâches et les transitions entre eux constituent la principale source d'information pour définir l'état courant d'un modèle de tâches. En outre, la considération des contraintes temporelles des tâches (durée minimale, durée maximale, ...) est importante pour assister l'utilisateur (par exemple en lui conseillant d'essayer d'être plus rapide dans une tâche B afin de rattraper le temps perdu sur la tâche A). Les algorithmes de suivi doivent également être adaptés pour pouvoir prendre en compte ces contraintes temporelles plus riches et plus fortes dans le cas des tâches quotidiennes (laver le linge, prendre ses médicaments à des instants/fréquences donnés, préparer le dîner, arroser les plantes, nettoyer l'aquarium...).

Plutôt que de proposer un nouveau modèle de tâches ex nihilo, nous avons préféré partir d'un modèle existant et essayer de l'étendre pour répondre aux besoins de la modélisation des tâches dans un environnement ambiant. Après avoir étudié les différents modèles et vu la non disponibilité de CTML, nous avons choisi le modèle CTT qui sert de référence pour de nombreux travaux de recherche. En effet le modèle CTT offre : (1) un ensemble riche d'opérateurs temporels pouvant relier les différentes tâches qui constituent un modèle ; (2) la possibilité de définir les pré et post-conditions d'une tâche selon une certaine syntaxe (qu'on se propose d'affiner par la suite) ; (3) un éditeur qui permet de dessiner le modèle de tâches et de vérifier sa cohérence (CTT Environment ou CTTE) [Mori et al. 2002][Paterno 2002].

Table I. Tableau comparatif des différents modèles et outils de modélisation des tâches étudiés

Modèle et domaine	Opérateurs temporels	États dynamiques des tâches	Spécification des conditions	Spécification des performances temporelles	Outils	Types de tâches
K-MAD conception et évaluation ergonomique de logiciels interactifs	-séquentiel, alternatif, parallèle, pas d'ordre et élémentaire	-détermination de groupe de tâches faisables ou pouvant être démarrées -attribution d'états dynamiques aux tâches abstraites	-pré-conditions sur les objets manipulés -Conditions exprimées en logique du premier ordre	-durées précisées mais non exploitées -on peut les saisir mais les tâches sont considérées atomiques	-éditeur et simulateur K-MADe -éditeur et simulateur ProtoTask	-utilisateur -système -interactive -abstraite
CTT spécification de systèmes interactifs	-choix, ordre indépendant, parallèle, synchronisation, disabling, suspendre/reprendre, séquentiel	-considération des enabled Task Sets -une tâche est atomique	-conditions selon syntaxe spécifique manipulant les objets du domaine et évaluées	-précision des durées minimales maximales et moyennes mais non exploitées	-éditeur et simulateur CTTE	-utilisateur -système -interactive -abstraite
COMM conception de systèmes multi utilisateurs multi modaux	-opérateurs temporels de CTT -opérateurs d'Allen		-pré-conditions et effets renseignés dans un champ texte mais pas évalués		-éditeur e-Comm disponible en ligne	-système(2 types) -individuelle(3 types) -de groupe(3 types)
HAMSTERS support à la conception et au développement d'un système interactif	-opérateurs temporels de CTT	-tâche effectuée -tâche disponible -tâche désactivée	-conditions sur les objets	-durées minimales et maximales pour l'évaluation des performances temporelles	-outil logiciel disponible HAMSTERS	-utilisateur(3 types) -système -interactive(3 types) -abstraite
CTML support de la modélisation des tâches dans un domaine collaboratif	-opérateurs temporels de CTT		-conditions sur les objets du domaine et l'emplacement	-durées minimales et maximales pour l'évaluation des performances temporelles	-éditeur CTML non distribué	-ceux de CTT

4. PRÉSENTATION DE NOTRE MODÈLE DE TÂCHES

Dans cette section, nous allons décrire notre modèle de tâches, inspiré du modèle CTT et qui l'adapte aux spécificités des tâches réalisées dans des environnements ambiants. Par ailleurs, l'état courant du modèle en exécution est capable d'évoluer dynamiquement pour refléter les changements continus de l'environnement et de l'activité de l'utilisateur. Notre système de suivi de tâches au cours de l'exécution sera basé sur l'utilisation de ce nouveau modèle. Nous commencerons cette section par décrire une transformation initiale que nous appliquons au modèle dans l'unique but de simplifier les algorithmes qui seront décrits par la suite.

4.1 Construction de l'arbre des tâches

Partant d'un modèle de tâches CTT, nous commençons par le transformer de façon à obtenir un modèle de tâches sous forme d'un arbre binaire [2] sans liens horizontaux entre les tâches : les liens horizontaux de CTT, qui représentent les opérateurs temporels, sont remplacés par de nouvelles tâches abstraites intermédiaires correspondant à ces opérateurs. Cette transformation en arbre binaire est introduite dans le but de simplifier l'algorithme de suivi qui n'aura plus qu'à s'appliquer sur un arbre binaire classique sans devoir gérer en plus des liens horizontaux entre les nœuds de l'arbre. Nous distinguons deux types de tâches dans notre arbre binaire : les tâches concrètes et les tâches abstraites. Les tâches concrètes sont les feuilles de l'arbre. Elles représentent des tâches qui peuvent être détectées par les capteurs de l'environnement, lors de leur exécution par l'utilisateur, le système ou bien les deux dans le cas de tâches interactives. Ces tâches représentent le niveau de décomposition le plus bas. Une tâche abstraite définit la logique d'ordonnancement de ses tâches filles. Elle représente le comportement d'un opérateur temporel reliant deux sous-tâches, comme illustré sur la figure 1 .

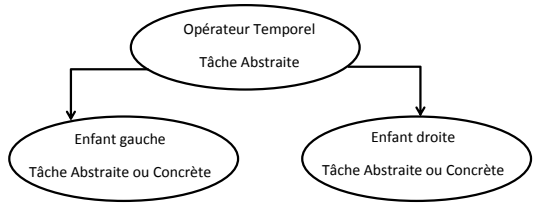


Fig. 1. Exemple de décomposition de tâches

La figure 2 montre un exemple de la façon dont un modèle CTT (conçu avec CTTE l'éditeur de CTT) est transformé en un arbre binaire.

4.2 Propriétés des tâches concrètes

Une tâche concrète a un certain nombre de propriétés qui peuvent être soit statiques (une fois établie lors de la phase de conception, la valeur de la propriété ne pourra plus être modifiée lors de la phase d'exécution), soit dynamiques (la valeur de la propriété peut évoluer au moment de l'exécution en fonction des informations provenant des capteurs).

2. Il est également possible d'utiliser un arbre n-aire.

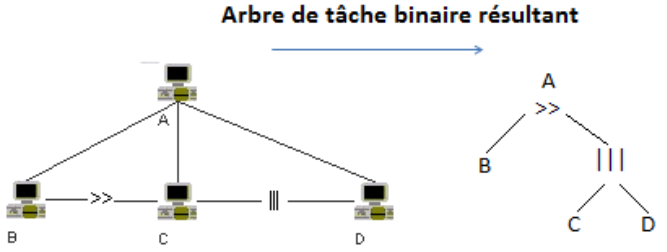


Fig. 2. Arbre de tâches binaire résultant de la transformation

Les propriétés statiques d'une tâche sont de deux types. Nous retrouvons certaines propriétés qui ont été déjà introduites dans CTT et qui sont :

- Son identifiant (son nom).
- Son importance (élevée, moyenne, basse) : cette caractéristique sera utile lorsque nous aurons besoin d'attribuer des priorités à des tâches concurrentes.
- Son caractère obligatoire : détermine si la réalisation de la tâche est indispensable pour atteindre un objectif donné. Dans certain cas la non-réalisation d'une tâche peut avoir des conséquences graves sur l'environnement (par exemple : couper l'arrivée du gaz) d'où le besoin de renseigner ce champ.
- Sa durée maximum de réalisation : renseigne sur la durée maximum prévue pour la réalisation de la tâche
- Sa durée minimum de réalisation : donne une information sur la durée minimum prévue pour la réalisation de la tâche

Nous aurons également un certain nombre de propriétés qui ont été affinées ou introduites pour les besoins propres de notre modèle et qui sont :

- Événement de démarrage et de fin : Nous associons à chaque tâche un événement de début et de fin qui permettent la détection du début de la réalisation de cette tâche et la fin de celle-ci.
- Service abstrait : représente une entité logicielle dont a besoin une tâche système pour s'exécuter. Notons que cette entité peut éventuellement être le fruit d'une composition de services préalable (cette dernière problématique n'est pas traitée dans ce travail). Pour les tâches purement utilisateur ce champ sera vide puisqu'il n'y aura pas de sollicitation de services logiciels.
- Pré-condition et post-condition : Une pré-condition est une condition qui doit être vérifiée pour que la tâche ne puisse se produire. Les pré-conditions d'une tâche incluent toutes les conditions nécessaires à sa réalisation comme par exemple : être dans un emplacement précis, manipuler un objet précis, etc.
  Les pré-conditions qui concernent les relations d'ordonnancement entre les tâches (par exemple, une tâche B ne peut commencer que lorsque la tâche A est terminée) ne sont pas représentées à l'aide des pré-conditions mais sont prises en compte au sein du modèle de tâches via les opérateurs temporels. En effet l'arbre de tâches CTT décrit les différents chemins possibles pour atteindre un but donné, et de ce fait il contient implicitement les pré-conditions temporelles relatives à l'ordonnancement des tâches. Les post-conditions constituent un ensemble de conditions dont on sait qu'elles doivent être vérifiées après l'exécution de la tâche. Elles décrivent l'effet

de la tâche sur l'environnement (ou plus précisément une partie de l'effet, utile au concepteur). Les pré/post conditions peuvent contenir une information sur principalement trois grandes catégories de composantes à savoir :
- l'environnement, par exemple la température, la luminosité...
- l'état des dispositifs du système
- l'utilisateur : ses profils statique et dynamique. Par exemple le sexe de l'utilisateur sera une caractéristique statique alors que la position de l'utilisateur sera une caractéristique dynamique puisqu'elle peut changer au cours de son interaction avec le système.

Les pré-et post-conditions qui sont déjà incluses dans le modèle CTT utilisent une syntaxe spécifique qui ne nous permet pas d'exprimer toutes les conditions possibles. Les pré-conditions utilisées par notre modèle sont écrites selon une syntaxe que nous avons définie et sont évaluées au moment de l'exécution. Les pré- et post-conditions que nous proposons actuellement sont des combinaisons logiques des données fournies par les capteurs, exprimées en logique propositionnelle. Elles combinent ces données avec des opérateurs logiques « OU », « ET » et « NON » pour permettre au concepteur d'exprimer des tests élémentaires sur les valeurs fournies par les capteurs. A titre d'exemple, une pré-condition peut se présenter comme ((P1 || P2) && P3) où P1, P2 et P3 représentent des propositions logiques sur l'état des capteurs et déclarées dans un fichier de configuration sous la forme de <proposition_logique> : <variable><opérateur_de_comparaison><variable/valeur>.

Les propriétés dynamiques d'une tâche que nous avons introduites sont :
- État de la tâche (inactive, possible, active, suspendue, réalisée, arrêtée) : déterminé en utilisant les relations temporelles reliant les différentes tâches (voir Figure 3).
- État des pré-conditions : modifié lorsqu'un événement se produit dans l'environnement.
- État des post-conditions : modifié suite à l'exécution d'une tâche.

Notons qu'en ce qui concerne les tâches abstraites, leurs propriétés sont déduites récursivement à partir de celles de leurs tâches filles et de la logique temporelle les reliant.

4.3 Les opérateurs temporels

Les opérateurs temporels utilisés dans notre modèle sont ceux de CTT. Nous rappelons dans ce qui suit les différents opérateurs, selon leur ordre de priorité, en commençant par le plus prioritaire (cet ordre est pris en compte lors de la transformation de l'arbre) :
- Choice [] : la tâche droite ou la tâche gauche peut avoir lieu (uniquement une des deux peut être réalisée).
- Order Independent |=| : les deux tâches doivent avoir lieu dans n'importe quel ordre, on peut commencer avec la réalisation de la tâche droite puis la gauche ou inversement.
- Interleaving ||| : les deux tâches sont réalisées en parallèle.
- Synchronization |[]| : les deux tâches doivent être synchronisées, elles démarrent et se terminent en même temps.
- Disabling [> : la tâche gauche peut être interrompue à tout moment par la tâche droite mais sa reprise ne sera plus possible.
- Suspend-Resume |> : la tâche gauche peut être interrompue à tout moment par la tâche droite qui permettra une fois terminée, la reprise de la tâche gauche.
- Sequential Enabling » : la tâche gauche doit être terminée pour que la tâche droite puisse commencer.

4.4 Les états des tâches

Comme nous proposons une approche qui utilise dynamiquement le modèle de tâches, l'état de toutes les tâches (nœuds et feuilles) est recalculé en continu en fonction des changements survenus dans l'environnement. Les états possibles d'une tâche sont :

- INACTIVE : état initial ; la tâche n'a pas encore été effectuée.
- POSSIBLE : signifie que la tâche peut être effectuée dès que ses pré-conditions sont satisfaites.
- RÉALISABLE : la tâche peut être effectuée (ses pré-conditions sont satisfaites).
- ACTIVE : la tâche est en cours d'exécution.
- ACTIVE-SUSPENDUE : cette tâche était en cours d'exécution (Active) lorsqu'une autre tâche a démarré et l'a suspendue. Sa réalisation pourra être reprise une fois que la nouvelle tâche sera terminée.
- POSSIBLE-SUSPENDUE et RÉALISABLE-SUSPENDUE : Le même cas de suspension s'applique pour une tâche pouvant être réalisée (Possible ou Réalisable) qui peut être suspendue (Possible-Suspendue ou Réalisable-Suspendue) et ne pourra être démarrée que lorsque la tâche qui l'a suspendue sera terminée.
- ARRÊTÉE : l'exécution de la tâche a été interrompue par une autre et ne pourra pas être reprise. Notons toutefois que contrairement à la suspension, l'arrêt d'une tâche ne peut s'appliquer qu'aux tâches qui ont été effectivement démarrées (Actives) et non aux tâches en attente de démarrage.
- RÉALISÉE : la tâche a été réalisée entièrement avec succès.

Les états des tâches et les transitions entre eux sont représentés dans la figure 3.

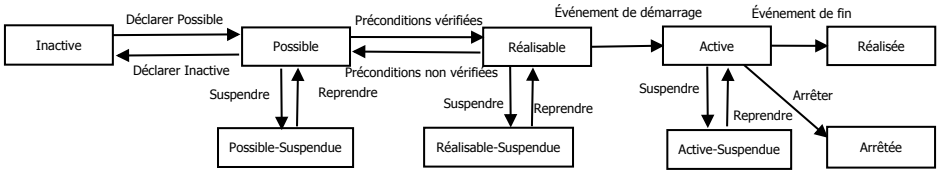


Fig. 3. Les états possibles d'une tâche sous forme d'une machine à état finis

Par défaut, une tâche est à l'état « INACTIVE ». Les états des différentes tâches sont déterminés chaque fois que le système reçoit un événement de l'environnement. Une tâche « RÉALISABLE » passe à l'état actif lorsque le système constate le début de la réalisation de celle-ci. Une fois que l'événement de fin qui reflète l'achèvement de la tâche est reçu, le modèle de tâches modifie l'état de la tâche à « RÉALISÉE ». Le cas d'une tâche répétitive est pris en charge au niveau du modèle de tâches, en l'occurrence dans notre cas, lors de l'édition du modèle de tâches (sur CTTE c'est une case à cocher). Si une tâche est répétitive elle pourra repasser par tous ces états.

4.5 Les états des pré-conditions et des post-conditions

Les états des pré-conditions et des post-conditions des tâches sont liés aux valeurs retournées par les capteurs. Les pré-conditions permettent de déterminer si une tâche est réalisable. Si une tâche s'est déroulée avec succès les valeurs retournées par les capteurs devraient alors valider ses post-conditions (par exemple « Allumer une lampe » devrait augmenter le taux de luminosité renvoyer par le capteur situé à proximité). Notons qu'une

post-condition relative à une tâche peut être à son tour une pré-condition pour une autre tâche. Une propagation à l'ensemble des autres tâches où ces post-conditions apparaissent en tant que pré-conditions est alors nécessaire pour vérifier l'éventuelle satisfaction de leurs pré-conditions.

5. UN SYSTÈME D'AIDE ET DE SUIVI BASÉ SUR LE MODÈLE DE TÂCHES

Cette section est dédiée à la présentation de notre système d'aide et de suivi qui exploite dynamiquement les données contenues dans le modèle de tâches (décrivant ce qui devrait être fait). La logique de fonctionnement de ce système repose sur l'exploitation des performances temporelles prévues et effectives. En effet, cette propriété (la durée des tâches) se prête bien à des méthodes quantifiables. Nous détaillons dans ce qui suit la stratégie d'intervention de ce système et son principe d'intervention que nous illustrons à travers un scénario de cuisine.

5.1 Un scénario dans un environnement ambiant

Afin de faciliter la compréhension de notre approche, nous présentons un scénario que nous utiliserons dans la suite de l'article :

"Il est 10h00. Jean s'apprête à recevoir des amis dans 2 heures. Ayant le choix entre commencer par la préparation du déjeuner ou par le nettoyage de la maison, il décide de commencer par préparer à manger.
Il commence par la cuisson des pâtes : il prend tout d'abord la casserole dans le placard, la rempli d'eau et la met sur le feu. Dix minutes plus tard, une fois que l'eau est bouillante, il met les pâtes dans la casserole, les laisse cuire pendant 15 minutes avant de les égoutter. Pendant que les pâtes sont en train de cuire, il prépare la sauce en commençant par éplucher un oignon. À ce moment, le téléphone fixe qui est dans le salon sonne. Jean suspend sa tâche en cours pour répondre à l'appel. Une fois la communication terminée, il revient à la cuisine, coupe l'oignon en petits morceaux, prend la poêle dans le placard et y met l'oignon. Il ajoute ensuite l'huile et met le mélange dans la cocotte puis cinq minutes plus tard y verse le contenu d'une sauce préalablement achetée en grande surface et laisse cuire pendant dix minutes. Dix minutes plus tard le système ayant détecté que la plaque de cuisson n'a pas été éteinte, il indique alors à Jean d'enlever la sauce de la plaque de cuisson et d'éteindre cette dernière. Jean mélange ensuite les pâtes avec la sauce. Une fois le déjeuner prêt, il passe au nettoyage de la maison."

5.2 Architecture globale de la solution

Dans une étude précédente nous avons présenté un système de suivi et d'assistance [Gharsellaoui et al. 2013] qui utilise le modèle de tâches (présenté ci-dessus) au moment de l'exécution des tâches par l'utilisateur en tant que référence à ce qui doit être fait pour atteindre un but particulier. Le principe de notre approche est de comparer, au moment de l'exécution des tâches, les informations contenues dans ce modèle à l'activité réelle de l'utilisateur afin de pouvoir détecter un besoin d'assistance. La figure 4 illustre cette approche.

Le système de supervision (3) reçoit en permanence des informations issues des capteurs (2) sur ce qui se passe réellement dans l'environnement. Ces informations servent à mettre à jour d'une part l'état des tâches (détection du démarrage et de la fin d'une tâche donnée) et d'autre part l'état des pré-conditions. Les valeurs reçues des capteurs influencent directement les valeurs des pré-conditions qui les utilisent comme des variables. Dans notre

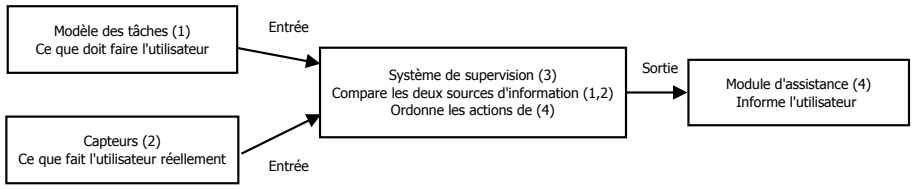


Fig. 4. Approche globale

scénario, à titre d'exemple, la tâche « Sortir poêle du placard » a comme pré-condition « le placard est ouvert ». Cette pré-condition est vérifiée par le capteur posé au niveau de la porte du placard. La détection du bon déroulement des tâches est quant à elle effectuée grâce à la vérification des états des post-conditions affectées aux tâches à travers les valeurs retournées par les capteurs. La vérification de ces post-conditions indique que la tâche concernée a eu l'effet souhaité sur l'environnement. Par exemple la vérification du bon déroulement de la tâche « éteindre plaque de cuisson » revient à recevoir une information de baisse de température du capteur de température thermique de la plaque de cuisson. Ces informations sont alors comparées à celles contenues dans le modèle de tâches (1) qui décrit ce que l'utilisateur devrait faire pour atteindre un but donné. Si le système constate que l'utilisateur est en train de faire une action différente de ce qui a été prévu dans le modèle ou que une tâche n'a pas eu l'effet souhaité sur l'environnement, il peut décider d'appeler le module d'assistance (4) afin d'informer l'utilisateur que telle sous-tâche a été omise ou qu'elle n'a pas été faite correctement. Par exemple, dans le cas où l'utilisateur oublie d'éteindre la plaque, le système après avoir détecter ce dysfonctionnement, appelle le module d'assistance qui rappellera à l'utilisateur de faire cette tâche.

5.3 Stratégies du système de supervision

Le système de supervision repose sur l'utilisation d'un module logiciel, le contrôleur de tâches, dont le but est de contrôler le déroulement correct des tâches. Ce contrôleur de tâches est lui-même composé de deux modules de contrôle différents ayant chacun un rôle particulier :

- Le contrôleur de tâches concrètes ou CTC : le CTC s'occupe du contrôle des tâches concrètes. En utilisant les données issues des capteurs, il surveille le déroulement des tâches qui se trouvent au niveau des feuilles de l'arbre des tâches selon un algorithme commun à toutes les tâches concrètes.
- Le contrôleur de tâches abstraites ou CTA : le CTA gère le contrôle de l'ordonnancement des tâches. La stratégie de contrôle du CTA est spécifiée par des algorithmes différents selon le type de tâche abstraite, car chaque type de tâche abstraite correspond à un opérateur temporel différent.

Avant d'aborder la description détaillée de ces deux contrôleurs il est nécessaire de définir une notion importante : le cycle de vie nominal d'une tâche.

5.3.1 *Cycle de vie nominal d'une tâche.* La figure 5 détaille le cycle de vie nominal d'une tâche (sans suspension ou arrêt). Par défaut toutes les tâches sont à l'état « Inactive ». L'état « Possible » est affecté à une tâche en fonction des relations temporelles définies entre les différentes tâches constituant le modèle (par exemple suite à la fin d'exécution de la tâche qui la précède). Ensuite la tâche devient « Réalisable » une fois que toutes ses pré-

conditions sont vérifiées. Enfin elle passe à l'état « Active » à la réception d'un événement externe indiquant le commencement de sa réalisation. Une fois que l'exécution de la tâche s'achève, le système reçoit un événement externe indiquant que la tâche est terminée. Son statut devient alors « Réalisée ».

« Le temps d'attente maximum d'activation » est défini comme étant la durée maximale prévue entre le moment où la tâche devient possible et le moment où commence concrètement sa réalisation. Cette durée de temps tient compte du « temps nécessaire pour la satisfaction des pré-conditions » indispensable pour la réalisation de la tâche et d'« une marge de tolérance » (définie par le concepteur) après laquelle le démarrage de la réalisation de la tâche devient urgent. Enfin « le temps de réalisation » correspond à la durée maximale prévisionnelle dédiée à la réalisation de la tâche. La figure 5 illustre ces différents états et les temps les séparant, décrivant ainsi l'ordonnancement nominal des états d'une tâche qui va servir de référence pour le suivi de celle-ci.

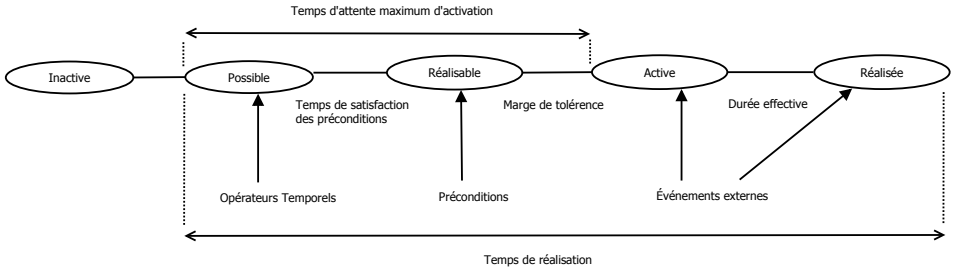


Fig. 5. Cycle de vie nominal d'une tâche

5.3.2 *Le CTC.* Le CTC « surveille » les actions de l'utilisateur (à partir des données issues des capteurs) et appelle le module d'assistance dans un des deux cas suivants :

(1) l'utilisateur fait une tâche qu'il ne devrait pas faire : il s'agit du cas où l'utilisateur entame la réalisation d'une tâche alors que ce n'est pas le bon moment de la faire.

(2) l'utilisateur fait la bonne tâche mais pas dans les bonnes conditions : il s'agit du cas où l'utilisateur entame la réalisation d'une tâche alors que ses pré-conditions ne sont pas vérifiées (la localisation par exemple) ou encore ne manipule pas la (les) ressource(s) nécessaire(s).

(3) l'utilisateur fait la bonne tâche mais pas comme il faut : il s'agit du cas où la réalisation de la tâche n'a pas eu l'effet attendu sur l'environnement.

Notons qu'il existe un quatrième cas où le module d'assistance peut être appelé (« l'utilisateur ne fait pas la tâche qu'il devrait faire ») mais ce cas est pris en charge par le CTA et sera présenté plus tard. Dans ce qui suit nous allons détailler chacun des trois cas dans lesquels le CTC va solliciter l'intervention de l'assistant en s'appuyant sur les définitions ci-dessus.

(1) L'utilisateur fait une tâche qu'il ne devrait pas faire.

Les données issues des capteurs permettent au CTC d'inférer qu'un utilisateur est en train de faire une tâche qu'il ne devrait pas faire dans le cas où celui-ci entame la réalisation d'une tâche alors que ce n'est pas le moment de la faire. Dans ce cas, le CTC

reçoit un message indiquant que la tâche a été démarrée et qu'elle devrait donc passer à l'état « Active » alors que son état précédent est l'état « Inactive ». Par exemple, dans notre scénario ce cas peut arriver dans le cas où Jean commence à faire le ménage alors qu'il n'a pas fini de préparer à manger.

(2) L'utilisateur fait la bonne tâche mais pas dans les bonnes conditions. Le CTC peut conclure qu'une tâche n'est pas (ou n'a pas été) effectuée dans les bonnes conditions en se basant sur les deux critères suivants :
   – Vérification des pré-conditons :
     Les données issues des capteurs permettent également au CTC d'inférer qu'un utilisateur est en train de faire une tâche qu'il ne devrait pas faire dans le cas où celui-ci entame la réalisation d'une tâche alors que ce n'est pas le moment de la faire et/ou que ses pré-conditions ne sont pas vérifiées (tâche non réalisable). Dans ce cas, le CTC reçoit un message indiquant que la tâche a été démarrée et qu'elle devrait donc passer à l'état « Active » alors que son état précédent est l'état « Possible ». Par exemple, dans notre scénario ce cas peut arriver dans le cas où Jean essaye de mettre les pâtes dans l'eau alors qu'elle n'est pas encore bouillante.
   – Les modalités de réalisation :
     Une tâche peut nécessiter obligatoirement la manipulation d'une ou plusieurs ressources matérielles. Ceci implique auparavant une catégorisation des appareils selon les services qu'ils offrent. Si l'utilisateur est en train de réaliser une tâche, il doit forcément manipuler une ressource appartenant à la catégorie de ressources que nécessite la tâche. Dans le cas contraire, on peut dire que l'utilisateur est en train de réaliser la tâche mais pas de la manière attendue. Par exemple, le système peut détecter ce cas si Jean utilise le micro-onde pour la cuisson et pas la gazinière (puisque la poêle et la casserole ne peuvent pas être introduites dans le micro-onde ainsi que certains aliments).

(3) L'utilisateur fait la bonne tâche mais pas comme il faut.

   Le CTC peut conclure qu'une tâche n'est pas (ou n'a pas été) effectuée correctement en se basant sur l'effet de la réalisation de cette tâche sur l'environnent :

   Les post-conditions constituent un ensemble de conditions qui doivent être vérifiées une fois la tâche exécutée. Elles décrivent son effet sur l'environnement. Une tâche réalisée mais qui ne produit pas les effets attendus sera donc considérée comme une tâche n'ayant pas été effectuée correctement (voir section 5.2). Le CTC va donc vérifier si les post-conditions sont bien satisfaites pour les tâches interactives et utilisateurs. Ce cas peut se présenter dans notre exemple dans le cas où ou le système détecte que l'utilisateur a effectué la tâche « éteindre la plaque de cuisson » alors que celle-ci reste à température élevée après une certaine durée (grâce à un capteur de température). Pour les tâches système on suppose que celles-ci sont toujours bien réalisées par le système. Le cas où les tâches système sont susceptibles de ne pas être réalisées correctement a été traité dans le cadre d'un autre travail de thèse [Mohamed 2013].

5.3.3 *Le CTA.* Le CTA a en charge la supervision temporelle globale des tâches selon deux axes :

– l'instant d'activation (correspond au cas où l'utilisateur ne fait pas une tâche qu'il devrait faire). Dans le scénario, cela se produit si l'utilisateur dépasse 10 minutes après l'ébullition de l'eau et ne commence pas la tâche de « cuisson des pâtes ».

– la durée de réalisation. Ce cas peut arriver dans le scénario si la tâche de « cuisson des pâtes » dépasse les 15 minutes comme prescrit.

En ce qui concerne le contrôle de l'instant d'activation, le CTA informera l'utilisateur qu'il a du retard sur le démarrage de la tâche si le temps d'attente maximum d'activation qui lui a été associé est simplement dépassé alors que la tâche est toujours dans l'état « Possible » ou « Réalisable » (voir section 5.3.1). Pour le contrôle de la durée de réalisation, il s'agit d'un processus plus complexe. Nous expliquerons dans un premier temps, son fonctionnement dans le cas de tâches séquentielles et nous montrerons ensuite comment ce fonctionnement peut être étendu dans le cas des autres types d'opérateurs temporels.

La durée de réalisation

On définit la durée de réalisation d'une tâche t comme étant la durée séparant l'instant où cette tâche est déclarée comme étant « Possible » jusqu'à l'instant ou celle-ci atteint l'état « Réalisée ».

Ici le CTA va s'intéresser à la durée de temps mise pour la réalisation de toutes les tâches de l'arbre. Pour cela on considère les deux fonctions $D_{\mathrm{MAX}}(t)$ et $D_{\mathrm{eff}}(t)$, qui définissent respectivement la durée maximale de la tâche t prévue par le concepteur, et la durée effective mise pour accomplir la tâche. Deux cas peuvent alors se présenter :

(1) soit la tâche est réalisée dans un laps de temps inférieur ou égal à la durée maximale qui lui a été attribuée ($D_{\mathrm{MAX}}(t) \geq D_{\mathrm{eff}}(t)$) : dans ce cas on disposera d'une marge égale à $D_{\mathrm{MAX}}(t) - D_{\mathrm{eff}}(t)$.

(2) soit la tâche est réalisée dans un laps de temps supérieur à la durée maximale qui lui a été attribuée ($D_{\mathrm{MAX}}(t) < D_{\mathrm{eff}}(t)$) : dans ce cas on signalera un retard sur la tâche t de durée $D_{\mathrm{eff}}(t) - D_{\mathrm{MAX}}(t)$.

Si le cas 2 se présente, l'utilisateur a du retard par rapport à la durée prévue pour la réalisation de cette tâche. Le problème consiste à déterminer le moment où le CTA devra appeler le module d'assistance pour alerter l'utilisateur sur l'existence d'un risque de ne pas pouvoir terminer la réalisation des tâches dans le délai de temps restant. Considérons dans un premier temps, un arbre de tâches composé uniquement de n tâches séquentielles T_1 à T_n. En plus des contraintes temporelles que le concepteur peut spécifier au niveau de chaque tâche concrète, il est possible de poser une contrainte temporelle globale sur l'ensemble des tâches à réaliser (comme représenté sur la figure 6). La valeur $D_{\mathrm{MAX}}(t)$ d'une tâche mère t qui se décompose en un ensemble de tâches séquentielles peut être obtenue de deux manières différentes :

– spécifiée par le concepteur : le concepteur décide d'attribuer une durée de temps à un ensemble de tâches. On peut imaginer par exemple que l'utilisateur ne doive pas mettre plus d'une heure pour se préparer le matin avant d'aller travailler. Ainsi, en plus des contraintes temporelles sur les différentes sous-tâches (prendre sa douche, prendre son petit-déjeuner, etc.) il est possible de spécifier une durée maximale totale de 60 minutes.
  Cette durée totale doit vérifier la condition suivante : $(\sum_{t' \in C} D_{\mathrm{MAX}}(t')) \leq D_{\mathrm{MAX}}(t)$, avec C l'ensemble des tâches concrètes situées sous la tâche mère t.
– le cas échéant calculée selon un cumul des durées prévisionnelles de ses tâches concrètes comme suit : $D_{\mathrm{MAX}}(t) = \sum_{t' \in C} D_{\mathrm{MAX}}(t')$;

Comme indiqué précédemment, le problème consiste à déterminer le moment où le CTA devra appeler le module d'assistance pour alerter l'utilisateur sur l'existence d'un risque de ne pas pouvoir terminer la réalisation des tâches dans le délai de temps restant. Les

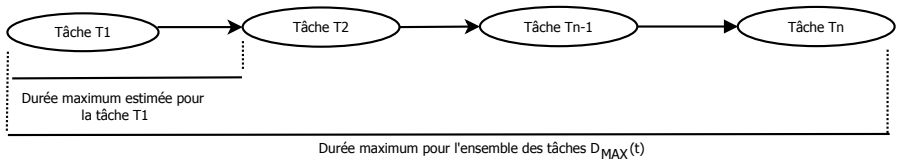


Fig. 6. Temps maximum attribué à un ensemble de tâches

données de notre problème sont les suivantes :

- n tâches à réaliser de façon séquentielle
- $D_{\mathrm{MAX}}(i)$ pour i allant de 1 à n : durée maximum prévisionnelle pour la réalisation de la tâche i
- $D_{\mathrm{MIN}}(i)$ pour i allant de 1 à n : durée minimum prévisionnelle pour la réalisation de la tâche i
- t_0 : temps où la tâche 1 (première tâche de la chaîne) passe à l'état possible
- t_{c} : temps courant
- k : nombre de tâches réalisées à l'instant t_c
- $D_{\mathrm{eff}}(i)$ pour i allant de 1 à k : durée effective mise pour la réalisation de la tâche i

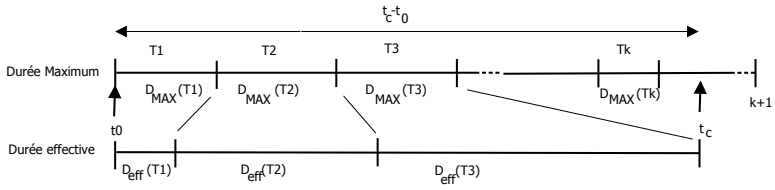


Fig. 7. Temps maximum et effectifs d'un ensemble de tâches

Le CTA construit un tableau temporaire (cf. Table II) pour l'ensemble des tâches de la chaîne qui contiendra les temps de début, ainsi que les durées maximales et minimales prévisionnelles d'exécution des tâches. Pour calculer l'instant de début au plus tard d'une tâche, il se réfère aux durées maximales d'exécution $D_{\mathrm{MAX}}(i)$ et pour calculer les instants de fin au plus tard il se réfère aux durées minimales d'exécution $D_{\mathrm{MIN}}(i)$. Les instants de début sont calculés en partant de la première tâche de la chaîne alors que les instants de fin sont eux calculés à partir de la dernière tâche de la chaîne.

Tâche	Début au plus tard	Fin au plus tard
T_1	0	...
T_2	$D_{\mathrm{MAX}}(1)$	...
T_3	$D_{\mathrm{MAX}}(1) + D_{\mathrm{MAX}}(2)$	...
...	...	...
T_k	$D_{\mathrm{MAX}}(1) + ... + D_{\mathrm{MAX}}(k)$	$D_{\mathrm{MAX}} - ... - D_{\mathrm{MAX}}(k+1)$
...	...	...
T_{n-1}	$D_{\mathrm{MAX}}(1) + ... + D_{\mathrm{MAX}}(n-2)$	$D_{\mathrm{MAX}} - D_{\mathrm{MIN}}(n)$
T_n	$D_{\mathrm{MAX}}(1) + ... + D_{MAX}(n-1)$	D_{MAX}

Table II. Calcul des instants de début et de fin au plus tard

Nous explicitons ci-dessous l'algorithme du contrôleur de tâches abstraites :

(1) calculer la durée $t_c - t_0$

(2) repérer dans le tableau la tâche attendue T_{att} en se basant sur les temps de début au plus tard. Notons i_{att} son indice. La tâche attendue correspondra à celle ayant l'instant de début le plus grand entre les tâches ayant un instant de début au plus tard inférieure à $t_c - t_0$

(3) deux cas de figures sont possibles :

   (a) cas 1 : $k \geq i_{att}$ signifie que l'utilisateur est en avance par rapport au planning prévu : l'intervention du système n'est pas nécessaire

   (b) cas 2 : $k < i_{att}$ signifie que l'utilisateur est en retard par rapport au planning prévu :

      i. cas 2a : $(\sum_{j=k+1}^{n} D_{\text{MIN}}(j)) \leq (D_{\text{MAX}} - (t_c - t_0))$, en d'autres termes la durée de temps restante peut suffire pour finir les tâches de $(k+1)$ à n si on se base sur les durées maximales des tâches.

      ii. cas 2b : $(\sum_{j=k+1}^{n} D_{\text{MIN}}(j)) > (D_{\text{MAX}} - (t_c - t_0))$, en d'autres termes la durée de temps restante ne suffira pas pour finir les tâches de $(k+1)$ à n si on se base sur les durées minimales des tâches.

Les stratégies d'intervention

Deux stratégies d'intervention peuvent alors être considérées :

(1) Stratégie intrusive ou stratégie pessimiste : dès qu'il y a un dépassement des durées maximales définies par le concepteur le système doit produire des alertes, autrement dit, dès que le cas 2 se présente on décide de produire des alertes pour l'utilisateur.

(2) Stratégie moins intrusive ou stratégie optimiste : nous considérons que bien que l'utilisateur ait pris du retard sur la réalisation des tâches de 1 à k, la situation peut éventuellement être rattrapée. Le système vérifie d'abord si le temps restant peut suffire pour la réalisation des tâches restantes dans l'hypothèse où l'utilisateur ait des performances maximales (en s'appuyant sur les durées minimales). Le système ne génère donc des alertes à l'utilisateur que dans le cas 2b.

Généralisation aux autres opérateurs

Nous venons de voir la stratégie d'intervention du contrôleur de tâches abstraites appliquée à un ensemble de tâches séquentielles. Nous allons voir maintenant comment ceci s'étend aux autres types d'opérateurs temporels. Pour les opérateurs temporels restants nous distinguons deux classes d'opérateurs :

(1) Un opérateur temporel à une seule possibilité de réalisation : l'opérateur « Synchronization » (voir section 4.3). Il existe une seule manière possible pour réaliser les deux tâches : démarrer leur réalisation en même temps et la finir en même temps.

(2) Les opérateurs temporels avec un choix qui se fait à l'exécution, qui eux-mêmes peuvent être partagés en deux groupes :

   (a) ceux qui ont leurs deux tâches filles obligatoires : « Interleaving » et « Order Independent ».

   (b) ceux qui n'ont qu'une tâche obligatoire, l'autre tâche étant optionnelle (pas forcément réalisée) qui à leur tour peuvent être partagés en deux groupes :

i. nous ne connaissons pas par avance la tâche qui sera réalisée : cas de l'opérateur « Choice »

ii. nous connaissons par avance quelle tâche sera réalisée (partiellement ou entièrement mais on est sûr de commencer avec cette tâche) : « Suspend Resume » et « Disabling ».

Pour les calculs des temps de début et de fin au plus tard sur lesquels va se baser notre système de supervision, il faut propager les calculs sur notre structure en arbre, pour cela on peut utiliser deux stratégies possibles :

(1) Stratégie 1 : Tenir compte des tâches optionnelles. Nous faisons les calculs des temps de début et de fin au plus tard avec les durées des tâches mères en faisant des calculs sur les pires cas et meilleurs cas de réalisation.

(2) Stratégie 2 : Ne pas tenir compte des tâches optionnelles. Nous faisons les calculs des temps de début et de fin au plus tard en tenant compte uniquement des durées des tâches qui ont forcément lieu à l'exécution. Nous mettons à jour les calculs en tenant compte des durées des tâches optionnelles si elles ont lieu concrètement (par exemple la tâche principale « Préparer à manger » peut être interrompue par « Un appel téléphonique ») et mettre à jour le calcul pour les tâches qui suivent.

Pour les opérateurs temporels différents de « Sequential » nous raisonnons sur notre arbre de tâches et nous faisons des appels récursifs de tâche mère à tâche fille. Nous commençons par attribuer à la tâche racine comme temps de début au plus tard l'instant 0 et comme temps de fin au plus tard la durée maximale attribuée à l'ensemble des tâches. Ensuite, nous lançons le calcul des temps de début et de fin au plus tard pour toutes les tâches en les propageant d'une tâche mère vers ses tâches filles jusqu'au niveau des tâches concrètes. Chaque tâche mère qui gère le comportement d'un opérateur temporel va procéder au calcul d'une manière spécifique. Dans le cas où les tâches sont répétitives, avec un nombre indéfini de répétitions, il n'est pas possible de déterminer la durée maximale par avance. Le tableau III résume la stratégie 1 sur les temps prévisionnels.

5.4 Principe d'intervention du système d'assistance

Intéressons nous maintenant aux stratégies d'assistance à l'utilisateur au cours de la réalisation de ses tâches. Dans le cas où le système est appelé à assister l'utilisateur, il doit choisir une ou plusieurs tâches vers lesquelles il va l'orienter. Au niveau de chaque tâche abstraite le système d'assistance détermine (de manière dynamique) une tâche candidate qui est la tâche vers laquelle il va orienter l'utilisateur. Tous les cas possibles correspondant à une tâche abstraite peuvent être classés selon trois groupes :

(1) Cas 1 : Une tâche abstraite décrivant le comportement d'un opérateur « Sequential », « Disabling » et « Suspend-Resume » proposera la tâche « enfant gauche ».

(2) Cas 2 : Une tâche abstraite décrivant le comportement d'un opérateur « Synchronization » proposera ses deux tâches enfants « enfant gauche et droite ».

(3) Cas 3 : Une tâche abstraite décrivant le comportement d'un opérateur « Interleaving », « Order Independent » et « Choice » proposera de démarrer avec une des deux tâches enfants « enfant gauche ou droite ».

Dans le troisième cas, pour proposer une tâche candidate il peut y avoir plusieurs tâches possibles et le système doit guider l'utilisateur vers l'une de ces tâches. Pour cela, l'en-

Opérateur	Début au plus tard	Fin au plus tard
Sequential		
Parent	durée max = somme des durées max enfants	durée min= somme des durées min enfants
Fils Gauche	début parent	fin parent - durée min frère droite
Fils Droite	début parent + durée max frère Gauche	fin parent
Synchronization		
Parent	durée max = max des durées enfants	durée min= min des durées min enfants
Fils Gauche	début parent	fin parent
Fils Droite	début parent	fin parent
Interleaving		
Parent	durée max = somme des durées max enfants	durée min= somme des durées min enfants
Fils Gauche	début parent	fin parent
Fils Droite	début parent	fin parent
Order Independent		
Parent	durée max = somme des durées max enfants	durée min= somme des durées min enfants
Fils Gauche	début parent	fin parent
Fils Droite	début parent	fin parent
Choice		
Parent	durée max = max des durées max enfants	durée min= min des durées min enfants
Fils Gauche	début parent	fin parent
Fils Droite	début parent	fin parent
Suspend-Resume		
Parent	durée max = somme des durées max enfants	durée min= durée min enfant gauche
Fils Gauche	début parent+ Durée max frère Gauche	fin parent
Fils Droite	début parent	fin parent
Disabling		
Parent	durée max = somme des durées max enfants	durée min= min durée min (enfant gauche, enfant droite)
Fils Gauche	début parent	fin parent - durée min frère droite
Fils Droite	début parent + Durée max frère Gauche	fin parent

Table III. Tableau des temps prévisionnels de la stratégie 1

semble des pré-conditions de chaque tâche (qui vont conditionner la facilité de sa réalisation) vont être considérées. Nous supposons que le système dispose d'une fonction qui renvoie une note dynamiquement calculée relative à la facilité de réalisation des pré-conditions dans le contexte actuel [3]. Cette note relative à la pré-condition d'indice i de la tâche j est notée $NotePrec_{i,j}$; elle est normalisée entre 0 et 1 :

– 0 irréalisable dans le contexte actuel.
– 1 déjà vérifiée.

3. Par exemple, si la précondition porte sur un emplacement requis pour l'accomplissement de la tâche, la note affectée pourra dépendre de la distance d'éloignement de l'utilisateur par rapport à cet emplacement.

Le système associe une note à chaque tâche possible en fonction de :

(1) la note attribuée à ses pré-conditions calculée comme suit :
$NotePrécondTâchePossible_j = \sqrt[n]{\prod_{i=1}^{n} NotePrec_{i,j}}$; avec n le nombre de pré-conditions de la tâche j.
Cette formule (moyenne géométrique) a pour avantage de permettre de donner la note 0 à toute tâche i qui a une pré-condition irréalisable dans le contexte actuel.

(2) la note attribuée à sa durée de réalisation est calculée comme suit :
$NoteDuréeTâchePossible_j = \frac{DuréeTâcheMere}{DuréeTâcheCandidate}$

$NoteTâchePossible_j = \sqrt{NotePrécondTâchePossible_j \times NoteDuréeTâchePossible_j}$

La tâche candidate proposée par une tâche abstraite sera :
- l'unique tâche possible gauche pour le cas 1 associée à sa note.
- les deux tâches pour le cas 2. Le nœud s'attribue de la même manière une note qui sera égale à $NoteTâchePossible_j = \sqrt{NoteTâcheGauche_j \times NoteTâcheDroite_j}$
- celle qui aura la note la plus grande pour le cas 3.

Un processus d'élection est ensuite effectué au niveau des tâches abstraites du bas vers le haut jusqu'à la racine de l'arbre. L'utilisateur sera orienté vers la (ou les) tâche(s) concrète(s) qui aura (auront) le vote du nœud racine de l'arbre comme illustré sur la figure 8 (c'est la tâche la plus facile ou la plus faisable dans le contexte actuel). Nous surlignons sur la figure en rouge les tâches concrètes possibles. Nous considérons à titre d'exemple l'opérateur temporel « Sequential » qui a les deux tâches filles A et B de durées respectives 3 et 4 minutes. La tâche A est déclarée comme étant possible et sa note (NT=1.16) est calculée en considérant les formules détaillées ci-dessus. Ensuite le nœud « Sequential » remonte sa tâche candidate accompagnée de sa note. Cela est représenté par la flèche reliant le nœud « Sequential » à son nœud parent « Choice ». Au niveau du nœud racine le vote final s'effectue et les tâches les plus faisables et faciles dans le contexte actuel de l'utilisateur sont les deux tâches E et G (avec la note 1.25).

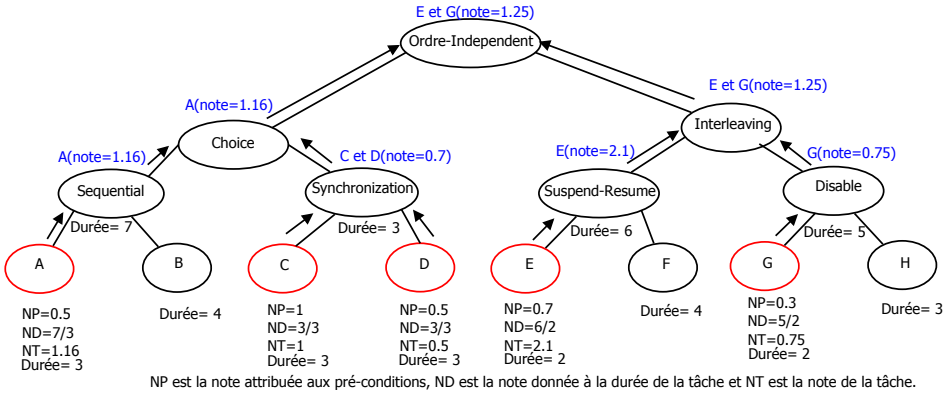


Fig. 8. Exemple de déroulement du processus d'élection d'une tâche candidate

6. SIMULATION D'UN SCÉNARIO

6.1 Le simulateur

Pour valider notre approche, nous avons développé un simulateur pour l'assistance à l'utilisateur et le suivi de ses tâches dans un environnement ambiant. Ce simulateur prend en entrée le fichier issu de la conception d'un modèle de tâches à l'aide de l'éditeur CTTE [Paterno 2002]. Cet outil peut donc être utilisé par le concepteur pour décrire son modèle de tâches comme illustré sur la figure 9. Le modèle résultant peut alors être intégré directement dans notre simulateur.

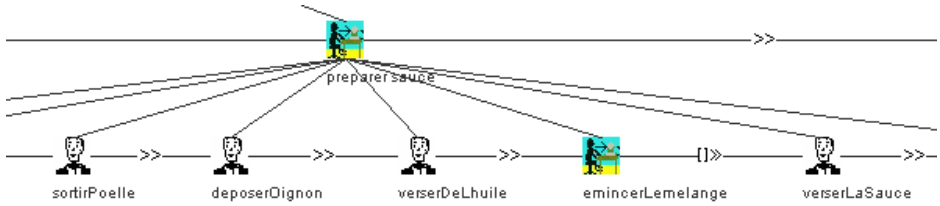


Fig. 9. Exemple d'une partie d'un arbre saisi avec CTTE

L'éditeur de CTT (CTTE) est utilisé pour saisir l'arbre de tâches. Pour chaque tâche, un ensemble d'informations est saisi (son identifiant, sa fréquence, ses durées maximum et minimum de réalisation,etc.). Par ailleurs, nous aurons besoin de définir pour chaque tâche un ensemble de propriétés qui ne sont pas prises en compte dans le modèle CTT (les services référencés, le temps maximum d'attente d'activation et l'ensemble de ses pré et post-conditions). Pour ajouter ces informations, le concepteur doit utiliser le champ de texte « description » dans CTTE. Les pré et post-conditions respectent une syntaxe que nous avons définie et qui est basée sur l'utilisation de la logique propositionnelle (chaque pré-condition représentant une proposition pouvant prendre les valeurs VRAI ou FAUX). Notre simulateur intègre un analyseur syntaxique qui effectue une analyse du fichier en entrée généré par CTTE. Pour chaque niveau de l'arbre de tâches, cet analyseur conserve les opérateurs temporels et restructure l'arbre de tâches en se référant aux priorités de ces opérateurs de manière à produire un arbre binaire sans liens horizontaux (voir section 3.1). Pour chaque tâche concrète, il stocke une copie des informations extraites à partir de l'arbre de tâches en entrée.

La figure 10 montre un exemple du modèle de tâches résultat sous forme d'un arbre binaire au sein du simulateur. Ce dernier permet de visualiser les opérateurs temporels reliant les différentes tâches concrètes ainsi que les informations les plus pertinentes concernant ces tâches et pouvant être utiles lors du déroulement de la simulation :

- le nom de la tâche ou son identifiant
- son état (calculé au moment de l'exécution)
- les informations relatives à ses pré-conditions :
  - la liste de ses pré-conditions et leurs états que le concepteur peut modifier au moment de la simulation.
  - l'urgence de l'activation de la tâche en fonction du délai restant par rapport au temps maximum d'attente d'activation qui lui a été prescrit au moment de la conception. Le voyant vert de la rubrique « Preconditions » indiquera qu'il reste

encore du temps pour la réalisation des pré-conditions et le voyant rouge s'allumera pour indiquer la nécessité de les satisfaire de manière urgente.

- les informations relatives à sa réalisation :
  - l'urgence de la réalisation de la tâche en fonction des alertes reçues de la part du contrôleur de tâches abstraites CTA. Le voyant vert de la rubrique « Realisation » indique qu'il reste encore du temps pour la réalisation de la tâche. Le voyant orange s'allume pour indiquer qu'il ne reste pas assez de temps pour la réalisation des tâches suivantes en s'appuyant sur une projection pessimiste c'est-à-dire en supposant que les tâches restantes consommeront la durée maximale spécifiée par le concepteur. Enfin le voyant rouge indique que le temps restant ne sera pas suffisant pour la réalisation des tâches restantes même en faisant une projection optimiste c'est-à-dire en considérant les durées minimales spécifiées par le concepteur.
  - la barre de progression se remplit en fonction du temps écoulé lors de la réalisation de la tâche (la durée effective de la tâche). La couleur de remplissage est bleue si la durée de réalisation qui a été prévue n'est pas encore dépassé et rouge dans le cas contraire.
  - un bouton « Start/ End » permet de simuler la détection du démarrage d'une tâche et celle de sa fin de réalisation.

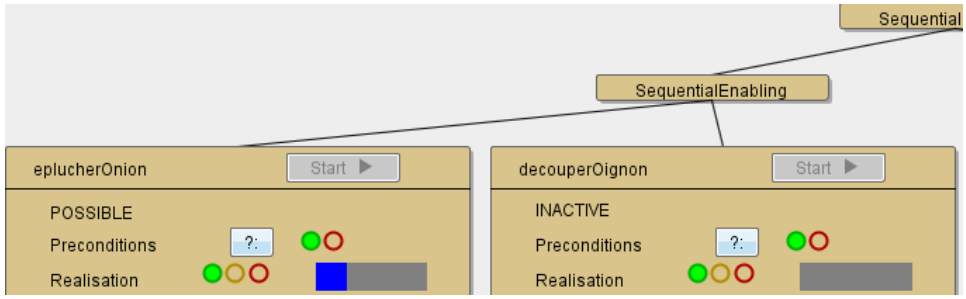


Fig. 10. Notre arbre de tâches binaire

6.2 Déroulement du scénario

Le simulateur que nous avons développé permet au concepteur de : (1) simuler la détection du démarrage d'une tâche et celle de sa fin de réalisation et (2) simuler à tout moment les modifications des états des pré-conditions d'une tâche donnée. Dans le système réel ces informations sont générées par le superviseur à partir des données reçues des vrais capteurs. Par exemple, la figure 11 illustre les pré-conditions de la tâche « SortirPoêle ».

Pour cette tâche les pré-conditions qui lui ont été assignées sont « placardOuvert » et « PoêleDansLePlacard » et sont reliées par un ET logique. Ceci signifie que la tâche ne peut être démarrée que si ces deux conditions sont satisfaites. Le concepteur peut simuler la satisfaction des pré-conditions en cochant les deux cases appropriées du simulateur. L'expression logique associée est alors évaluée par le simulateur et la faisabilité de cette tâche est donc calculée à l'exécution et elle devient réalisable comme sur la figure 12. Il est alors possible de cliquer sur bouton « Start » relatif à la tâche pour simuler la détection de son démarrage.

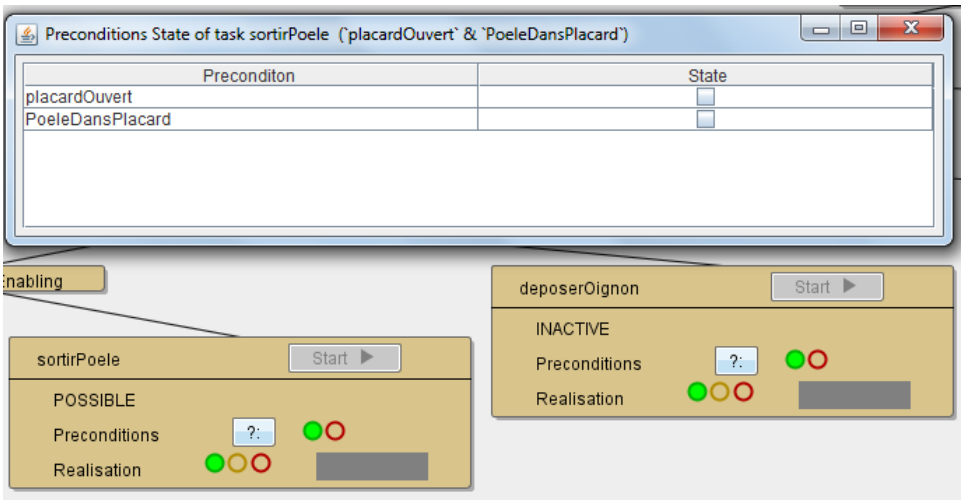


Fig. 11. Représentation des préconditions d'une tâche

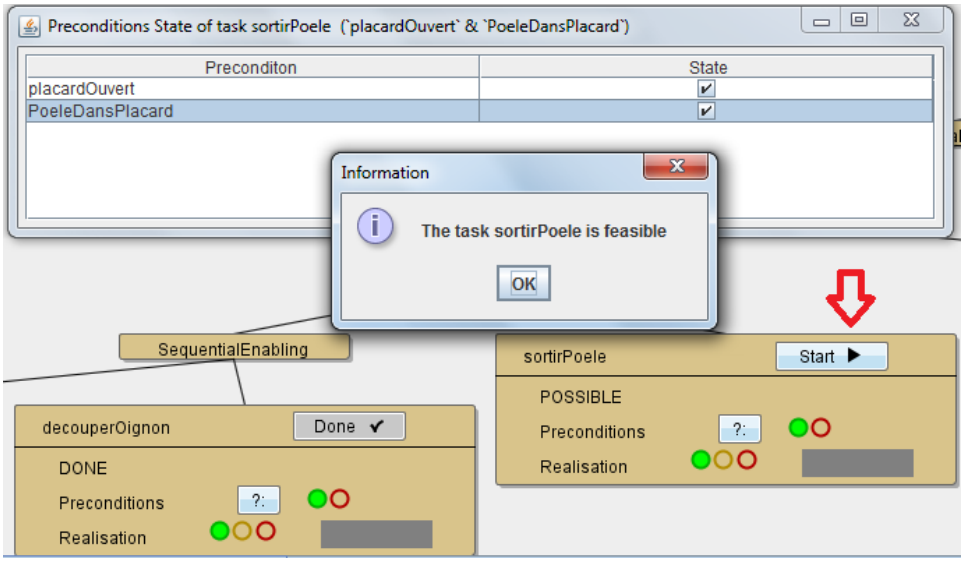


Fig. 12. Vérification des pré-conditions d'une tâche qui devient faisable

Une fois la tâche démarrée, l'étiquette du bouton « Start » devient « End » et le concepteur peut cliquer sur ce même bouton pour simuler la détection de la fin de la tâche (figure 13).

Supposons que l'utilisateur essaye de commencer la préparation des pâtes alors qu'il n'a pas encore mis en marche la gazinière. Le rôle du CTC peut être illustré par une notification sur le fait que la tâche est non réalisable comme le montre la figure 14 .

Afin d'illustrer le rôle du CTA, considérons la partie suivante du scénario : dès que l'utilisateur décroche le téléphone pour répondre à un appel sa tâche courante « éplucher

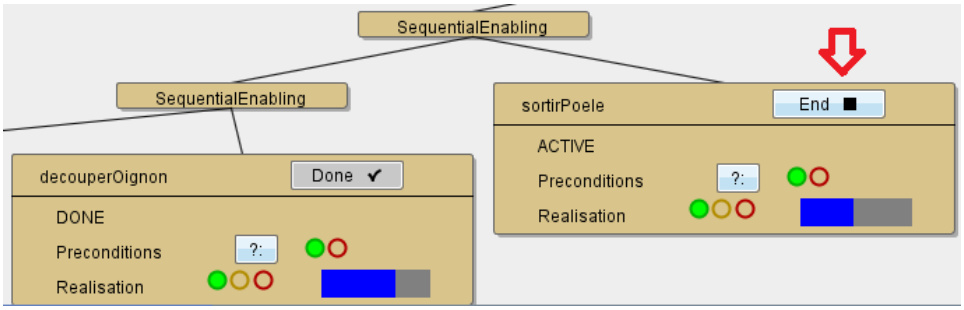


Fig. 13. Une tâche active peut être terminée

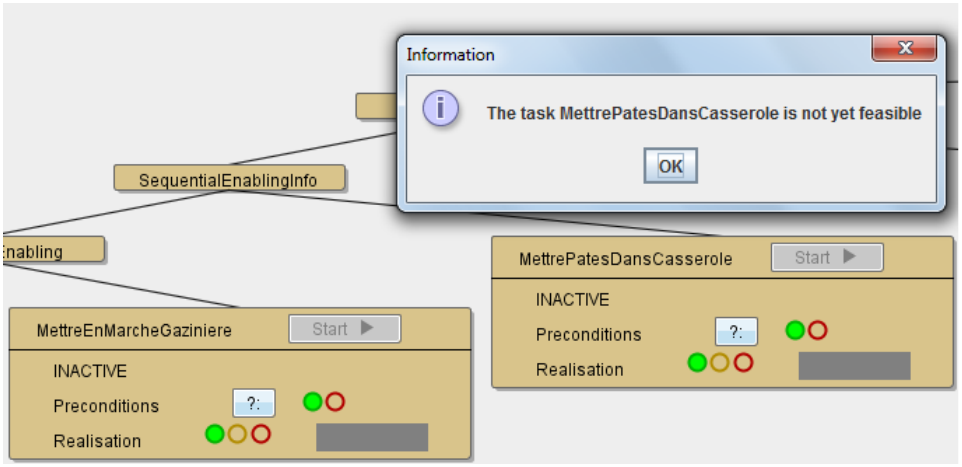


Fig. 14. le CTC notifie une tâche non faisable voulant démarrer

oignon » est suspendue. Pendant ce temps, le temps de réalisation de la tâche suspendue continue tout de même à avancer comme l'indique la figure 15.

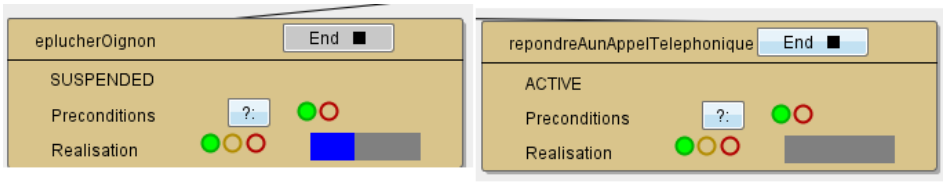


Fig. 15. L'activité en cours suspendue par un appel

Au moment où l'utilisateur finit de répondre à l'appel téléphonique sa tâche suspendue « éplucher oignon » peut être reprise. Comme le temps maximal prévu a été dépassé, la barre de progression est affichée en rouge sur la figure 16. De plus il ne restera plus de temps pour finir le reste des tâches même en les effectuant aux performances maximales

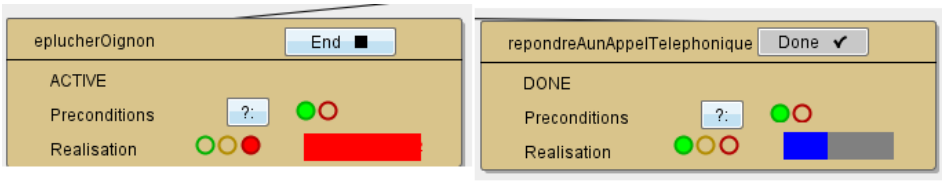


Fig. 16. Reprise de la tâche suspendue suite à la fin de l'appel et notification par le CTA du retard engendré

(avec les temps minimums prédéfinis). Par conséquent, le voyant rouge de réalisation s'allume comme indiqué sur la figure 16.

7. DISCUSSION

Une limite possible de notre approche concerne le fait que nous avons basé l'assistance sur la durée d'exécution des tâches. Plus particulièrement, dans le cas où les tâches sont répétitives, avec un nombre indéfini de répétitions, il n'est pas possible de déterminer la durée maximale de cette séquence. En effet, nous ne considérons que la durée de la première occurrence qui doit nécessairement avoir lieu pour calculer la durée de l'ensemble des tâches devant être effectuées. Une solution possible est de considérer une durée maximale pour toutes les répétions (le concepteur spécifie une durée maximale au niveau de la tâche mère). A titre d'exemple, si une tâche élémentaire est « faire une crêpe » qui peut prendre 5 minutes au maximum, nous pouvons considérer que l'utilisateur dispose de 2 heures au maximum pour « faire les crêpes ».

D'un autre côté, la qualité de l'assistance fournie par notre système repose en majeure partie sur la capacité du concepteur du modèle de tâches à spécifier des contraintes en conformité avec la « vie » effective des utilisateurs. Mais cela n'est pas toujours possible. A titre d'exemple nous pouvons citer les tâches domestiques ne sont pas toutes prévisibles. C'est pourquoi à plus long terme, nous envisageons également d'intégrer de l'apprentissage dans notre approche afin que le modèle de tâches puisse être enrichi et mis à jour au moment de l'exécution à partir des actions et des interactions avec l'utilisateur. Ainsi le système pourrait par exemple devenir capable d'apprendre une nouvelle façon de réaliser une tâche qui ne serait pas prévue au départ dans le modèle ou apprendre à ajuster certaines propriétés telles que les durées prévues pour cette tâche. Nous pensons qu'à terme, dans les environnements ambiants, l'utilisateur deviendra le concepteur de son propre système et à ce titre, doter le système de capacités d'apprentissage deviendra crucial.

Une autre amélioration possible de notre modèle de tâches concerne les pré-conditions, qui sont actuellement exprimées en logique propositionnelle, donc avec une expressivité limitée. Nous entendons utiliser le calcul des prédicats qui est plus complet ou encore la logique floue qui permet la modélisation des imperfections des données et se rapproche dans une certaine mesure de la flexibilité du raisonnement humain. La qualité des notifications produites par le système doit également être améliorée. Bien que l'information clé soit disponible à travers l'état des pré-conditions, il conviendra néanmoins de la traduire dans un langage qui soit compréhensible par l'utilisateur.

Enfin, pour offrir un plus grand pouvoir d'expression à notre modèle, nous envisageons de définir un opérateur temporel générique inspiré des relations temporelles d'Allen [Allen 1983] qui permettra d'établir toutes sortes de relations temporelles entre deux tâches. Pour exprimer les contraintes temporelles entre deux tâches A et B on considèrera les temps

de début et de fin de chacune. Ces temps seront reliés par des opérateurs de comparaison ($<, \leq, =, \geq, >$) pour constituer des expressions qui seront reliées à leur tour par des opérateurs logiques (OU, ET et NON). Cet opérateur se substituera aux opérateurs temporels de CTT tout en offrant un plus grand pouvoir d'expression. Par exemple dans CTT, l'opérateur « Synchronization » exprime uniquement la synchronisation parfaite des deux tâches reliées A et B (A et B démarrent et se terminent au même moment). Avec l'opérateur générique il sera possible d'exprimer différents types de synchronisation : synchronisation au démarrage uniquement (début A = début B), synchronisation à la fin uniquement (fin A = fin B) ou synchronisation parfaite de CTT (début A = début B ET fin A = fin B). En ce qui concerne les tâches séquentielles nous pourrons exprimer explicitement le délai entre la fin de la première tâche et le début de la suivante (si un laps de temps est requis entre les deux tâches).

8. CONCLUSION

Dans cet article, nous avons présenté un modèle de tâches adapté aux environnements ambiants, dont l'état peut être mis à jour au moment de l'exécution et qui étend la notation et la sémantique du modèle CTT (ConcreteTaskTree). Notre modèle permet d'attribuer des états aux tâches au moment de l'exécution en fonction des informations échangées avec l'environnement (démarrage d'une tâche, fin de réalisation d'une tâche, états des préconditions, etc.).

Nous avons également montré comment un système de suivi et d'assistance peut exploiter un tel modèle de tâches pour suivre au moment de l'exécution le déroulement des tâches, en fonction des états contenus dans le modèle de tâches et de ce qui se passe réellement dans l'environnement. Nous avons détaillé ensuite le fonctionnement du système de supervision et ce en définissant deux types de contrôleurs de tâche : un contrôleur de tâches concrètes chargé du contrôle de la logique d'exécution des tâches, et un contrôleur de tâches abstraites qui s'occupe plus particulièrement des aspects temporels liés aux tâches. Nous avons défini à quel moment chacun de ces deux contrôleurs doit intervenir afin d'alerter l'utilisateur en se basant sur différentes stratégies plus ou moins intrusives. Enfin nous avons détaillé le principe d'intervention du système d'assistance à l'utilisateur. Ce système détermine vers quelle tâche orienter l'utilisateur (si on dispose de plusieurs choix à un moment donné).

Enfin nous avons présenté une illustration de notre système à travers le déroulement d'un scénario sur notre simulateur. Cette simulation montre comment les interactions avec le modèle de tâches à l'exécution nous permettent de produire un système dynamique, qui prend en considération l'activité de l'utilisateur et lui fournit une aide pour la réalisation de ses tâches quotidiennes.

La prochaine étape importante de ce travail consistera à mener une évaluation expérimentale réelle dans la plate-forme du laboratoire dédiée à l'étude des environnements ambiants, afin de pouvoir tester l'apport d'un tel système auprès d'utilisateurs finaux. Cette expérimentation nous permettra de déterminer les meilleures stratégies d'interaction avec l'utilisateur afin de trouver le bon équilibre entre un système d'assistance trop discret et un système trop intrusif. Avec la croissance en Europe du nombre de personnes âgées, vivant seules ou atteintes d'Alzheimer, les systèmes d'assistance dans les environnements ambiants présentent des potentialités sociétales intéressantes pour apporter de l'aide à ces catégories d'utilisateurs. Ils offrent également des potentialités sur le plan économique en

aidant par exemple à réduire le temps post-opératoire que doit passer un malade à l'hôpital (e.g. aide à la prise de médicaments à domicile).

RÉFÉRENCES

ALLEN, J. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM 26,* 11 (Nov.), 832–843.

ANNETT, J. AND DUNCAN, K. 1967. Task analysis and training design. *Occupational Psychology 41*, 211–227.

BARON, M., LUCQUIAUD, V., AUTARD, D., AND SCAPIN, D. 2006. K-made : un environement pour le noyau du modele de description de l'activite. *18eme Conference Francophone sur l'Interaction Homme-Machine (IHM'2006)*, 287–288.

BARTHET, M. 1988. Logiciels interactifs et ergonomie : modèles et méthodes de conception. *Dunod*.

BLUMENDORF, M., LEHMANN, G., AND ALBAYRAK, S. 2010. Bridging models and systems at runtime to build adaptive user interfaces. In *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '10. ACM, New York, NY, USA, 9–18.

BOURGUIN, G., LEWANDOWSKI, A., AND TARBY, J.-C. 2007. Defining task oriented components. *Task Models and Diagrams for User Interface Design*, 170–183.

CARD, S., MORAN, T., AND NEWELL, A. 1983. The psychology of human-computer interaction. *Hillsdale , NJ : Erlbaum*.

DUCATEL, K., BOGDANOWICZ, M., SCAPOLO, F., LEIJTEN, J., AND BURGELMAN, J.-C. 2001. Scenarios for ambient intelligence in 2010. *Final report, Information Society Technologies Advisory Group (ISTAG)*.

FREY, A. G., CÉRET, E., DUPUY-CHESSA, S., CALVARY, G., AND GABILLON, Y. 2012. Usicomp : An extensible model-driven composer. In *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '12. ACM, New York, NY, USA, 263–268.

FRIEDEWALD, M. AND COSTA, O. D. 2003. Science and technology roadmapping : Ambient intelligence in everyday life (ami@life). IPTS Report 73.

GERHART, J. 1999. *Home Automation and Wiring*, 1 ed. McGraw-Hill/TAB Electronics. ISBN : 0070246742.

GHARSELLAOUI, A., BELLIK, Y., AND JACQUET, C. 2012. Requirements of task modeling in ambient intelligent environments. *International Conference on Ambient Computing, Applications, Services and Technologies*, 71–78.

GHARSELLAOUI, A., BELLIK, Y., AND JACQUET, C. 2013. A run time executable task model for ambient intelligent environments. *International Conference on Ubiquitous Intelligence (UIC 2013),International Workshop on Intelligent Techniques for Ubiquitous Systems (ITUS 2013)*, 691–696.

HARPER, R. 2003. *Inside the Smart Home*, 1 ed. Springer. ISBN : 1852336889.

JOHN, B. AND KIERAS, D. 1996. The goms family of user interface analysis techniques : Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 320–351.

JOHNSON, P. 1992. *Human-computer interaction : Psychology, task analysis and software engineering*. London : McGraw-Hill.

JOHNSON, P., DIAPER, D., AND LONG, J. 1984. Tasks, skill and knowledge : Task analysis for knowledge based descriptions. *In Proceedings of First IFIP Conference on Human-Computer Interaction (Interact '84) 1*, 23–28.

JOURDE, F., LAURILLAU, Y., AND NIGAY, L. 2009. Comm notation for specifying collaborative and multimodal interactive systems. *Proceedings of the second ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 125–134.

KLUG, T. AND KANGASHARJU, J. 2005. Executable task models. *TAMODIA 2005*.

LACHAUME, T., GIRARD, P., GUITTET, L., AND FOUSSE, A. 2012. Prototask, new task model simulator. 323–330.

LIMBOURG, Q. AND VANDERDONCKT, J. 2003. Comparing task models for user interface design. *The Handbook of Task Analysis for Human-Computer Interaction*, 135–154.

MAHFOUDHI, A., ABED, M., AND TABARY, D. 2001. From the formal specifications of user tasks to the automatic generation of the hci specifications. 331–347.

MARTINIE, C., NAVARRE, D., AND PALANQUE, P. 2014. A multi-formalism approach for model-based dynamic distribution of user interfaces of critical interactive systems. *International Journal of Human-Computer Studies 72,* 1, 77 – 99.

MARTINIE DE ALMEIDA, C., PALANQUE, P., BARBONI, E., WINCKLER, M. A., RAGOSTA, M., PASQUINI, A., AND LANZI, P. 2011. Formal Tasks and Systems Models as a Tool for Specifying and Assessing Automation Designs (regular paper). In *International Conference on Application and Theory of Automation in Command and Control Systems, Barcelone, 26/05/2011-27/05/2011*. IRIT Press, http ://www.irit.fr, (electronic medium).

MOHAMED, A. 2013. Fault-detection in ambient intelligence based on the modeling of physical effects. Ph.D. thesis, Supelec and LIMSI-CNRS.

MORI, G., PATERNO, F., AND SANTORO, C. 2002. Ctte : Support for developing and analyzing task models for interactive system design. *IEEE transactions on software engineering 28*, 797–813.

ORMEROD, T. C. AND SHEPHERD, A. 2004. Using task analysis for information requirements specification : The sub-goal template (sgt) method. 347–365.

PATERNO, F. 1999. *Model based design and evaluation of interactive applications*. Berlin : Springer-Verlag.

PATERNO, F. 2002. Task models in interactive software systems. In *Handbook of Software Engineering and Knowledge Engineering*. Vol. 1. World Scientific, 1–19.

PETOUD, I. 1990. Génération automatique de l'interface homme-machine d'une application de gestion hautement interactive. Ph.D. thesis, Université de Lausanne, Suisse.

PUNIE, Y. 2003. A social and technological view of ambient intelligence in everyday life : What bends the trend ? Technical Report EUR 20975 EN.

RIVA, G. 2005. The psychology of ambient intelligence : Activity, situation and presence. 17–33.

RIVA, G., LORETI, P., VATALARO, M., VATALARO, F., AND DAVIDE, F. 2003. Presence 2010 : The emergence of ambient intelligence. 60–81.

ROSCHER, D., LEHMANN, G., SCHWARTZE, V., BLUMENDORF, M., AND ALBAYRAK, S. 2011. Dynamic distribution and layouting of model-based user interfaces in smart environments. In *Model-Driven Development of Advanced User Interfaces*, H. Hussmann, G. Meixner, and D. Zuehlke, Eds. Studies in Computational Intelligence, vol. 340. Springer Berlin Heidelberg, 171–197.

SASSI, H., ROUILLARD, J., TARBY, J.-C., AND RENARD, G. 2010. Un système multi-agents permettant une assistance homme-machine mutuelle dans un environnement ambiant. *Quatrième Workshop sur les Agents Conversationnels Animés (WACA 2010)*.

SCAPIN, D. AND PIERRET-GOLBREICH, C. 1989. Towards a method for task description : Mad. 27–34.

SCHULUNGBAUM, E. 1996. Model-based user interface software tools current state of declarative models. *Visualization and Usability Center*.

SIOCHI, A. AND HARTSON, H. 1989. Task-oriented representation of asynchronous user interfaces. 183–188.

SOTTET, J.-S., CALVARY, G., COUTAZ, J., AND FAVRE, J.-M. 2008. A model-driven engineering approach for the usability of plastic user interfaces. In *Engineering Interactive Systems*, J. Gulliksen, M. Harning, P. Palanque, G. van der Veer, and J. Wesson, Eds. Lecture Notes in Computer Science, vol. 4940. Springer Berlin Heidelberg, 140–157.

STEPHANIDIS, C., PARAMYTHIS, A., SFYRAKIS, M., A. STERGIOU, N. M., LEVENTIS, A., PAPAROULIS, G., AND KARAGIANNIDIS, C. 1998. Adaptable and adaptive user interfaces for disabled users in the avanti project. In *Proceedings of the 5th International Conference on Intelligence and Services in Networks : Technology for Ubiquitous Telecom Services*. IS&N '98. Springer-Verlag, London, UK, UK, 153–166.

TARBY, J.-C. 1993. Gestion automatique de dialogue homme-machine à partir de spécifications fonctionnelles.

TARBY, J.-C. AND BARTHET, M.-F. 1996. The diane+ method. *Computer-aided design of user interfaces*, 95–120.

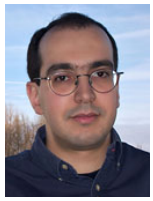
VANDERVEER, G. C., VANDERLENTING, B. F., AND BERGEVOET, B. A. J. 1996. Gta : Groupware task analysis - modeling complexity. *Acta Psychologica 91*, 297–322.

WEISER, M. 1991. The computer for the twenty-first century. 94–100.

WEISER, M. 1993. Some computer science issues in ubiquitous computing. *36,* 7, 75–84.

WURDEL, M., SINNIG, D., AND FORBRIG, P. 2009. Towards a formal task-based specification framework for collaborative environments. 221–232.

Asma Gharsellaoui est doctorante en informatique à l'Université Paris Sud 11 au sein du laboratoire de recherche LIMSI-CNRS et à Supélec. Elle est également monitrice d'informatique à l'IUT d'Orsay. Ses travaux de recherches portent sur la thématique de l'interaction Homme Machine dans les environnements ambiants. Elle travaille, plus particulièrement, sur l'étude des problèmes posés par la supervision des tâches utilisateur dans les environnements ambiants et la conception d'un modèle permettant le suivi et l'assistance des utilisateurs au moment de l'exécution de ces tâches.



Yacine Bellik est maître de conférences habilité à diriger des recherches. Il enseigne à l'université Paris-Sud (IUT d'Orsay) et mène ses recherches dans le laboratoire LIMSI du CNRS au sein duquel il dirige le groupe AMI (Architectures et Modèles pour l'Interaction). Ses recherches actuelles portent principalement sur l'interaction dans les environnements ambiants. Il s'intéresse également à la conception de systèmes interactifs multimodaux capables de s'adapter à différents contextes d'interaction grâce à l'exploitation de la multimodalité. Ses recherches trouvent des applications dans des domaines variés notamment dans celui du handicap visuel.

Christophe Jacquet est enseignant-chercheur à Supélec, sur le campus de Paris-Saclay. Il enseigne notamment la programmation, l'architecture des ordinateurs, les technologies du Web et les réseaux à Supélec, à l'École Centrale Paris et à l'École Polytechnique. Ses recherches actuelles portent sur les outils et les méthodologies de modélisation de systèmes hétérogènes qui combinent différents paradigmes de modélisation. Il s'intéresse aux applications de ces techniques dans les environnements ambiants.

Task model simulators: a review

Thomas Lachaume, Laurent Guittet, Allan Fousse, Patrick Girard, Allan Fousse

Task model simulators: a review

THOMAS LACHAUME
LAURENT GUITTET
PATRICK GIRARD
ALLAN FOUSSE
LIAS / ISAE-ENSMA – Université de Poitiers

Abstract: Task modelling has enabled the building of models of human activity for a long time. In the early years, pencil and paper where the only means available to build task models from task model notations. Because of the lack of computed constraints, task models often did not conform to the notation. To solve this problem, some tools were designed by authors in order to help users create, modify and save correct models that conform to the notation syntactic rules. However, understanding the full semantics of task models appeared difficult for practitioners. The dynamic aspects of task models could only be understood "in the user's head". New tools, named simulators, emerged to solve this problem. They allow to "run" or to "simulate" task models and to record scenarios. This execution fulfils the semantics of task model operators, which define the task dynamic semantics. Simulators can be used in many ways such as understanding model semantics, verifying or validating models, building valid scenarios, etc. In this article, we describe and compare currently available and maintained task model simulators, and explain the different usages of these tools, according to user goals and qualifications. Then, we explore the different challenges for these tools to exploit the complete semantics of task models.

Key words: Task model, Simulator.

Résumé : La description de l'activité humain-système par des modèles de tâches existe depuis plusieurs années. Après l'époque papier-crayon, les outils d'édition des modèles de tâches ont permis la conception et l'archivage des modèles selon une notation rigoureuse, et la vérification de leur cohérence. Mais la compréhension de la dynamique est restée affaire de spécialiste jusqu'à l'apparition des simulateurs de modèles. Une simulation permet d'appréhender les enchaînements réels de tâches - décrits implicitement par les opérateurs temporels - et de valider les scénarios ainsi réalisés.
Cet article décrit et compare les simulateurs actuellement disponibles et maintenus, et explique leurs différents usages en fonction des buts et niveaux d'expertise des utilisateurs. De nouvelles perspectives d'évolutions de ces modèles et outils sont alors définies dans le but d'améliorer leur sémantique.

Mots clés : Modèles de tâches, simulateurs.

Adresse des auteurs : <prenom.nom>@ensma.fr – LIAS / ISAE-ENSMA – Université de POITIERS – Téléport 2, 1 rue Clément Ader, BP 40109, 86961 Chasseneuil Futuroscope Cédex, France

1. INTRODUCTION

During the last forty years, task modelling has been an increasingly researched topic. Methods became more and more precise, and increased their expressive power by the way of new features, such as calculable expressions and objects. Several tools were designed to support the edition, understanding and validation of task models. Some success stories were reported (see for example [Paternò et al. 2012; Martinie et al. 2012]), while usability in real projects is becoming a major issue.

Nevertheless, task modelling still needs much involvement on the user's part in understanding the notations. Sometimes, personal interpretation of notations is required, despite the efforts made by their authors to make them unambiguous. The best way to eliminate ambiguities in interpreting task modelling notations should be to use the tools that are associated to the methods. These tools are supposed to cover all aspects of task modelling such as enforcing the syntactic rules of languages, helping model understanding, and illustrating their dynamic semantics[1].

This last point is performed by tools named simulators[2], which allow the user to interactively simulate the activity that is described by the task model. Unfortunately, no complete description is available for these tools, and when comparing them many differences can be observed in their layout and behaviour. More, some concepts, which are described in the notation, are not taken into account in the simulation. Last, in some cases, behaviours are not really coherent when different concepts are associated in models.

In this article, we present a comparative review of different simulators, chosen for their current availability. In section 2, we describe the state of the art of task modelling, and focus on common concepts and existing tools. In section 3, we review the behaviour of simulators, with two illustrative examples. Section 4 summarizes the different usages of simulators, and section 5 highlights the current challenges for task model simulators.

2. TASK MODELLING STATE OF THE ART

Several authors compared task modelling approaches and tools. Nevertheless, none of them really focused on simulators. In this section, we first give an overview of task modelling approaches, which evolved among the past forty years. Then, we introduce the basic principles of task modelling, and we survey the different task modelling tools. At the end of the section, we focus on simulators, and justify our choice of four systems to study in the remainder of this paper.

2.1. OVERVIEW OF TASK MODELLING APPROACHES

Task modelling emerged from attempts to understand human activity through activity analysis [Diaper 2004; Sebillotte and Scapin 1994]. As written in [Annett 2004], "analysis is not just a matter of listing the actions or the physical or cognitive processes involved in carrying out a task, although it is likely to refer to either or both. Analysis, as opposed to description, is a procedure aimed at identifying performance problems

[1] Beyond the syntactical aspects of models, the dynamic or behavioural semantics of models
[2] One could say that these tools mainly animate the models, and should be called "animators". In this article, we follow the common author naming practices.

(i.e., sources of error) and proposing solutions". Basically, two different approaches can be identified.

The first one addresses mainly predictive evaluation, and can be illustrated by GOMS. Goals, Operators, Methods, and Selection rules [John and Kieras 1996; Kieras 2004] leans on the Keystroke Level Model (KLM) [Card et al. 1983]. GOMS models provide a way to quantitatively predict human learning and performance for an interface design. It can be used to make a qualitative description of how the user will use a computer system to perform a task. Lots of extensions [Kieras 2004; John and Kieras 1996] and derived approaches such as UAN [Hix and Hartson 1993] were later developed.

The second approach can be illustrated by methods such as HTA (Hierarchical Task Analysis [Annett et al. 1971; Annett 2004], TKS (Task Knowledge Structure) [Johnson and Johnson 1991; Johnson et al. 1988], TAKD (Task Analysis for Knowledge Descriptions) [Diaper 1989], or MAD (Method for Analytical Description [Scapin and Pierret-Golbreich 1990; Sebillote 1992]). These methods assume that it is important for task analysis to describe not only the activities people carry out, but also the contexts in which they perform those activities, as well as the ways those activities are performed, and the tools and methods that performers use. Their primary purpose is to elicit the knowledge structures that skilled performers (usually) construct when performing tasks. Beyond evaluation, these methods can be used to support interactive system design. Further works explored these issues more deeply; they propose methods with associated tools, such as Diane+ [Tarby and Barthet 1996], Mad* ([Gamboa et al. 1997]) and TOOD (Task Object Oriented Design [Tabary et al. 2000]).

Because the real activity is not generally a single user activity, several methods focused on cooperative activity, such as GTA (Groupware Task Analysis [van der Veer 1996]), CTT (ConcurTaskTrees [Paternò et al. 1997; Paternò 1999]) and CTML (Collaborative Task Model Language [Wurdel et al. 2008]), or reliability and critical systems, such as VTMB (Visual Task Modelling Builder [M Biere et al. 1999]) or AMBOSS ([Giese et al. 2008]).

Research activity on task modelling remained active for the last ten years, with attempts to generalization, with K-MAD (Kernel of Model for Activity Description [Baron et al. 2006]) and USIXML [Limbourg et al. 2005], or standardisation (W3C[3]), as well as new formalisms associated with powerful tools (ECOMM [Jourde et al. 2010a], HAMSTERS [Martinie 2011]), aimed at providing efficient assistance in designing interactive systems.

2.2. BASIC PRINCIPLES OF TASK MODELS

Despite this large number of approaches and models, basic principles of task modelling remain very stable. Methods do not differ much from initial approaches, such as HTA, where tasks are defined in terms of goals and operations, are divided into subtasks that follow a strict hierarchy, and embed a "plan", which defines rules for subtask execution. Following a general tendency of formalisation in HCI, task modelling evolved towards more and more precision of manipulated concepts.

This section offers a short overview of the main concepts that are used by most task models. The use of such concepts is illustrated by an example: *"Take the train"*. This task aims at modelling the activity of a person who arrives at the railway station to

[3] http://www.w3.org/TR/2014/NOTE-task-models-20140408/

travel by train. The system is composed of a ticket vending machine, large displays providing departure announcements, and stamping machines. Depending on the circumstances, the traveller may already have a ticket, or, if not, may buy a ticket at the ticket window or at the ticket machine. The ticket may be a physical one, which requires the traveller to stamp it before boarding, or an electronic one, which does not require stamping. If the traveller has omitted stamping, the ticket can exceptionally be stamped in the train by the human controller. The traveller is supposed to watch the departure announcement panel to find his/her way. This example is illustrated with a CTT model (probably the most largely used task model today) in Figure 1, but other models with different formalisms are given in Annex.

Basics of task modelling approaches are hierarchical decomposition of tasks into subtasks, and characterization of said tasks. Models converged to the ability of distinguishing roles during activities, assuming that activities involve human beings and systems. *User tasks*, which involve only human beings, *system tasks*, which conversely involve only systems, and *interactive tasks*, which involve both, are the main categories. Generally, *Abstract tasks* are used in task hierarchy to figure out nodes that can be refined in different types of subtasks. In our example (Figure 1), *Take the Train* and *Get Ticket* are considered as Abstract tasks because they are complex tasks, which are broken down into tasks from different categories. *Get Ticket* is composed of two options, *Get Ticket at Ticket Window*, which is considered as a User task (it involves a relationship between two human beings), and Get Ticket at Ticket Machine, which is Interactive, because it involves the user and the system. *Find Train* is also split in subtasks, which are not abstract, because its two subtasks belong to the same interactive category.

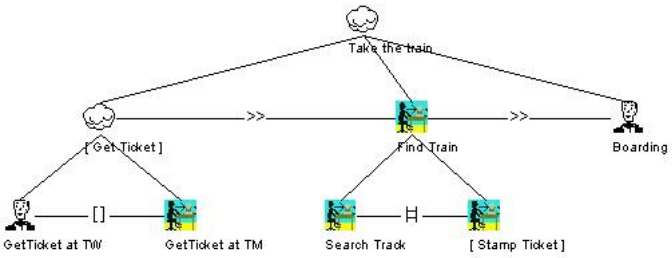


Figure 1: Take the train, CTT model

Annett's and Johnston's *Plans* between subtasks use quite equivalent ordering operators on every notation, which are based on a subset of LOTOS operators [Systems 1984; Paternò et al. 1992]. The basic Annett's plans, named *fixed sequence* (the activity is supposed to follow a predefined sequence of subtasks), *selective rule* (the user can choose between several subtasks), and *time-sharing* (subtasks are run concurrently) match respectively to *enabling* >>, *choice* [] and *concurrent* ||| operators. One more operator, *order independent* |=| (subtasks are not required to be made in a specific order), is generally added to this set. In our example, the first level of decomposition supposes a strict sequence between getting a ticket, searching for the train, and boarding. On the opposite, getting a ticket may be done at the ticket window or at the ticket machine, requiring a choice operator, while the *Find Train* task does not require any precedence between subtasks. Ordering operators are one of the major elements of task models, because they define the dynamics of the model.

Tasks own several attributes, which allow for a more precise description of the modelled activity. Some of them have consequences on task model dynamics, such as the optional attribute and the iteration attribute. The first one stands for optional tasks, which can be omitted during the activity. The second one defines repetitive tasks. In our model, *Get Ticket* and *Stamp Ticket* are optional. The first one needs to be done only if the user does not have previously bought a ticket. The second one is not required when e-tickets are used, and can be omitted if the traveller is late. In CTT, optional task names are represented between brackets.

Many task models and methods used several concepts to define operations that are the heart of activity. The need for context definition and for precision about conditions led to general notions of *precondition*, which defines the pre-requisite of the task, *action*, which describes the action to do to reach the goal of the task, and *post-condition*, which expresses the state required after the action. These elements can be informal (only texts are defined in the method) or formal; in that case, they involve objects and some kind of expression language. In our example, preconditions can be expressed on tasks, in order to state more precisely how they can be included in the activity. *Get Ticket* is subordinated to the absence of ticket, for example with a sentence like "*The traveller does not have ticket*" or with a condition on a Boolean object (*hasTicket* is ***false***). In the same way, *Stamp Ticket* is also guarded by a condition on the ticket (only possible if having physical ticket), and *Boarding* is guarded by the fact the traveller owns a ticket. Post-condition can be illustrated by an expression, which can be attached to *Get Ticket*: "*Assume that the traveller owns a ticket, either physical or electronic*". Actions can be described relatively to objects; for example, *Get Ticket at the TW* should be described with the action "The traveller buys a physical ticket" or *hasTicket* := ***true***[4]. With these elements, it is possible to verify the activity correctness by evaluating the pre/post-conditions, improving the global semantics of task models.

2.3. OVERVIEW OF TASK MODELLING TOOLS

Because modelling requires precision and non-ambiguity, many tools were designed to help task-modelling practitioners. Most tools are editors, which are able to enforce the notation used in the method. We can split these tools in two categories.

- Tools from the first category were designed after publishing the method, in order to help using it. The GOMS family owns several tools, such as compared in [Baumeister et al. 2000], and even first methods were supported by tools, such as ADEPT [Johnson et al. 1993] for TKS, Task Architect [Stuart and Penn 2004] for HTA, EMAD [Delouis and Pierret 1991] and ALACIE [Gamboa et al. 1997] for MAD and MAD*, or EUTERPE [van Welie et al. 1998] for GTA.
- Starting with CTT/CTTE [Paternò et al. 2001; Mori et al. 2002] the new generation of tools is deeply associated to the method and the notation from the early beginning. We can mention in this second category AMBOSS [Giese et al. 2008], TOOD/ETOOD [Tabary and Abed 2002], VTMB [M. Biere et al. 1999], and more recently K-MAD/K-MADe[5], e-COMM [Jourde et al. 2010b] and HAMSTERS[6].

[4] Which means: the Boolean object whose name is "hasTicket" becomes ***true***
[5] http://lisi-forge.ensma.fr/forge/projects/kmade
[6] http://www.irit.fr/recherches/ICS/softwares/hamsters/

All tools allow users to build task models, with respect to model rules. Most often, tools do no implement the whole task model, some rules being kept away.

Some tools stand on the dynamic semantics of notations to propose a specific category of tools, which allow the user to interactively simulate the modelled activity. Again, two main categories can be observed: simulators that require watching the model itself to simulate it, and simulators that do not display the model during the simulation (ProtoTask [Lachaume, Girard, et al. 2012]). We discuss more in depth this point in section 4.

We restricted our study to task model tools that fulfil the following requirements:

- Including a simulator that can illustrate most concepts of the underlying model
- Being widely available today, by free download or free registration
- Being supported, to allow bug correction and evolutions

Only three families of models/tools meet these requirements: CTT/CTTE, HAMSTERS, and K-MAD/K-MADe/ProtoTask. ProtoTask is part of the K-MADe environment, but must be considered as a specific simulator, very different from the standard K-MADe simulator. At our knowledge, the other systems are no longer available, or do not provide simulators.

3. COMPARISON OF SIMULATORS

Based on the same principles, all task model simulators work in a similar manner. Users are guided by task model semantics to simulate an activity, which follows strictly the task model dynamic behaviour. Beyond the obvious difference of visualisation, we can define some common principles that lead the simulation. Nonetheless, depending on the prime objective of the tool's authors towards modelling, some differences appear when carefully inspecting simulator capabilities.

The difference in task model visualisation is more than a simple choice between layout strategies, or icons to display tasks. It results from essential design choices, which lean on different principles.

The dynamic behaviour is the richest part of the models. Once again, choices that depend from authors' objectives lead to diverse solutions, which result in diverse possibilities for modelling. They result in the definition of constraints on the models.

In this section, we carefully study the solutions each system proposes. We reuse our first example, "Taking the train", and to illustrate some specific topics that cannot be explained on this example, we bring up the classical example of the "Cash machine", well-known in most HCI works. The "Cash Machine" is fully described in the Annex.

We start by describing basic simulator principles, which are all used by the different tools. Then, we detail the feedback the different simulators are able to provide to users. Last, we explain the simulator behaviour during simulation. In the last two sections, a table will summarize the studied criteria.

3. 1. SIMULATOR PRINCIPLES

Simulators are supposed to illustrate the concrete semantics of task models, as expressed by using task attributes and ordering operators. From the point of view of simulator's users, two basic functionalities can be brought out:

- interactive simulation of activities,
- building scenarios.

3.1.1. Interactive simulation of activities

The first objective of simulators is to show a concrete illustration of a simulated activity, "running[7]" task model semantics. At simulation run-time, subtasks are proposed to the user according to the different operators attached to the model. Similar to a debugger tool, the simulator works step-by-step, each step simulating one task; after each step, the user is asked to choose the next task. For example, in Figure 2, we can see the CTTE simulator during a cash withdraw task, while the user already entered his/her password and the required amount, and just before recovering his/her bank card. At this point in the task, the only possible action to do is *Withdraw Card*, because of the ***Enable*** operator (>>) between the two tasks. The only choice for the user is to accept the proposed task.

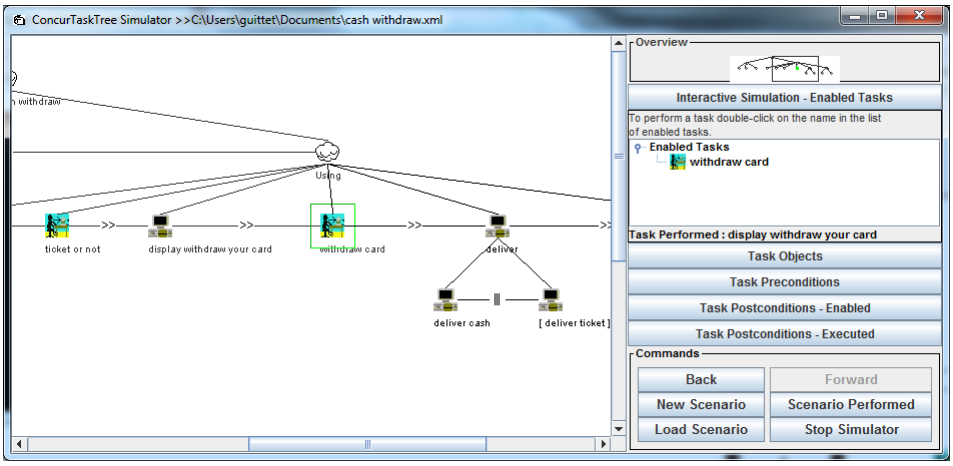


Figure 2: View of the CTTE simulator in action

Figure 3 shows the same simulator several steps later, when cash and ticket are available for the user, and *Withdraw Cash* and *Withdraw Ticket* are available at the same time because of the ***OrderIndependance*** operator (|=|). At this time, the user of the simulator is given the choice of selecting the order in which s/he wants these two tasks to be performed.

[7] in the computer science sense of the word

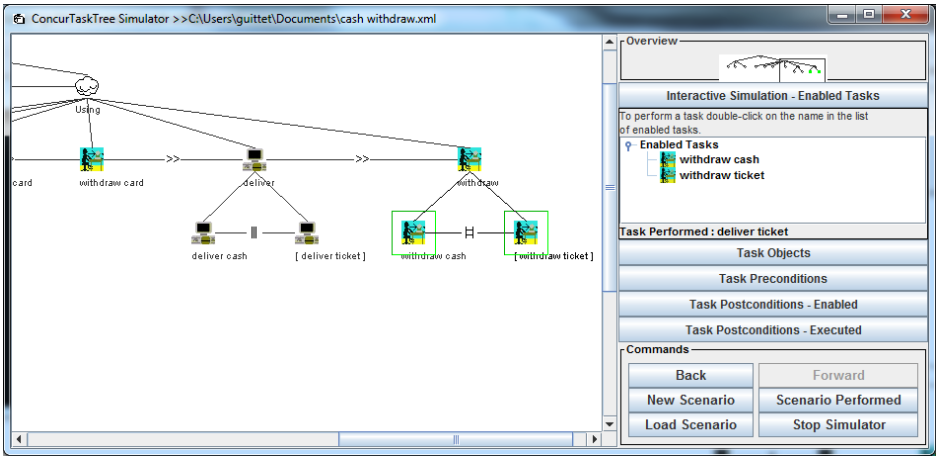


Figure 3: a simulator with several choices

While "running" the task model, the simulator builds a scenario, which stands for operating a concrete activity, in line with the task model. This scenario can be saved, and reloaded, in order to play it again onto the simulator for this particular task model.

The heart of simulators is the Enabled Task Set (ETS) calculus, which calculates all possible tasks within the model at each step [Paternò 1999]. In this definition *Enabling* means making a task available for simulation, alone or with all other available tasks conforming to task model operators.

Attributes can also be taken into account by simulators. For example, the ***Optional*** attribute states that a task can be omitted. So, when included in an ETS, this task does not prevent nor enforce the execution of other tasks. For example, if task *Withdraw Ticket* is defined as ***Optional*** (because the user does not need the ticket) the optional task can be skipped and the simulator goes to the next available task in the model.

Another attribute acts like an operator. It is the ***Iterative*** attribute. Attached to a task, it means that this task is in fact a repetitive task, whose simulation does not just end after all its subtasks have been done once.

The last versions of simulators take into account objects and/or pre/post-conditions to enable tasks. Preconditions, which can be based on objects, may guard task enabling. For example, the *Withdraw ticket* task should not really be optional in the point of view of the user. It should depend on the choice the user made several steps beforehand, when s/he decided whether s/he wanted a ticket. Without using preconditions, scenarios can be built that are not practical or accurate, for example by choosing first the option of not asking for ticket, and then attempting to withdraw a ticket. Preventing this kind of bad use of models can be made via the use of pre/post-conditions. The most general way to make pre/post-conditions in task models supposes the definition of objects (Boolean variables for example) to be attached to tasks as a precondition in the form of an expression, which can be evaluated by the simulator to state whether to enable the task. For example in this case, the *Withdraw Ticket* task can be guarded by a Boolean value whose meaning could be "Ticket required". We explain later how this function is implemented differently in each of the tools.

Each task model tool has its own specificities, depending on specific objectives followed by its authors.

CTT focuses on cooperation, with cooperative tasks, and on generation possibilities. It proposes more precise descriptions of objects, and introduces the platform the application is supposed to work on. HAMSTERS addresses "modelling in the large", with modularization features, and interactive animation with a link to the PetShop tool [Bastide et al. 2002]. K-MAD (including both K-MADe and ProtoTask) manages actors, improving the expressive power of task models.

All these topics induce specific management in the simulator tools. Because they all are specific to one tool, we did not include them in this review. We also did not specifically study the simulator's management of iterative tasks. In fact, the iteration itself is managed the same way in all tools. The difference in managing the end of iteration depends completely on the choice made in the notation: in CTT and HAMSTERS, ending iterations need the usage of a specific operator, called "disabling", while K-MAD provides a computer language like mechanism based on a loop condition.

3.1.2. Building scenarios

The second objective of simulators consists in building scenarios. While "running" a task model, the user builds a list of tasks, which is in fact an example of a concrete activity. All simulators are able to record in a file such scenarios, and to load and "run" again these scenarios. These scenarios do not generally include any value for objects, but only elementary tasks.

Beyond the above main principles, simulators differ in the following two ways. First, they display different information, depending on their global philosophy. Second, their behaviour during the simulation may be different.

3. 2. SIMULATORS LAYOUT

The main two simulator feedbacks concern the way tasks are proposed to the user, and the layout of current simulation. At the end of the section, we propose a table, which summarizes the differences between the systems.

3.2.1. Displaying the Enabled Task Set

The main objective of simulators is to allow the user to simulate, step-by-step, task models. At each step, the user is required to choose what task to do among the ETS. For this purpose, the system must give the user all information required to understand the task context in order to make an informed choice on the next task to perform. The only common view between all simulators is the ETS view, which allow the user to check what task can be run at each step.

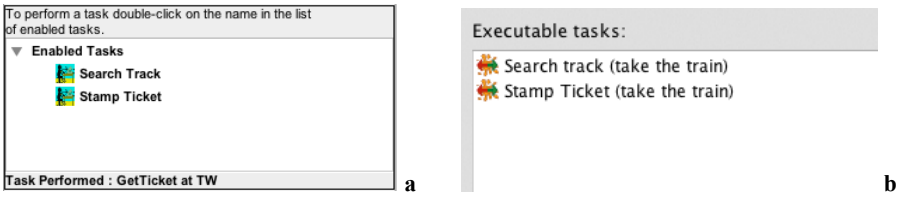


Figure 4: ETS in CTTE(a) and HAMSTERS(b)

HAMSTERS and CTTE display a list of enabled tasks, with no more information (Figure 4). K-MADe adds a qualification to each task, to make explicit the fact that the

user wants to "skip" the task. This point relates to optional tasks, whose execution depends on a choice made by the user. Figure 5a illustrates a state where a user can choose to stamp a ticket or not.

While only terminal tasks (the leaves of the task tree) are displayed in the first three simulators, the Prototask simulator chooses to represent things differently. First, it allows the selection of nodes as well as terminal tasks. Then, in a panel where all tasks from the same decomposition level are shown, only enabled tasks appear as enabled buttons while other tasks are visualized with disabled buttons. In the example of Figure 5b, we can see the ETS of the "Take the train" task model, after simulation of task *Get Ticket*. The other two tasks of the main level of decomposition are visualized, but greyed because they are not currently enabled (*Get Ticket* is done, and *Boarding* not yet enabled). We can also see that preconditions are also written near the task name.

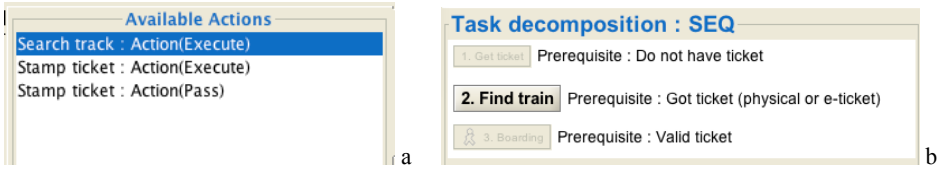


Figure 5: ETS in K-MADe(a) and Prototask(b)

3.2.2. Understanding the goal/subgoal hierarchy

To make the user understand the context of the current task, displaying the ETS is not enough. Because task trees reflect goal/subgoal mental decomposition of users, displaying only enabled tasks with no reference to the goal hierarchy would lead to an overflow of the cognitive load on the user's part. Two different strategies were used to solve this problem. The first strategy leans on the task model itself. It can be illustrated by the CTTE, HAMSTERS, and K-MADe simulators. In Figure 6, the task tree in CTTE uses the main part of the window, with colour codes to help understand what tasks are enabled: the simulation step is inside the *Find Train* task, after completion of task *Search Train*; conforming operator semantics, the only enabled tasks are *Stamp Ticket* (on the same level with *Search Train*, and an ***OrderIndependence*** operator), and *Boarding* (because *Stamp Ticket* is an optional task). All two are rounded by a green square on the graphic panel. Only ETS is added to the visualization, and buttons to control the simulation.

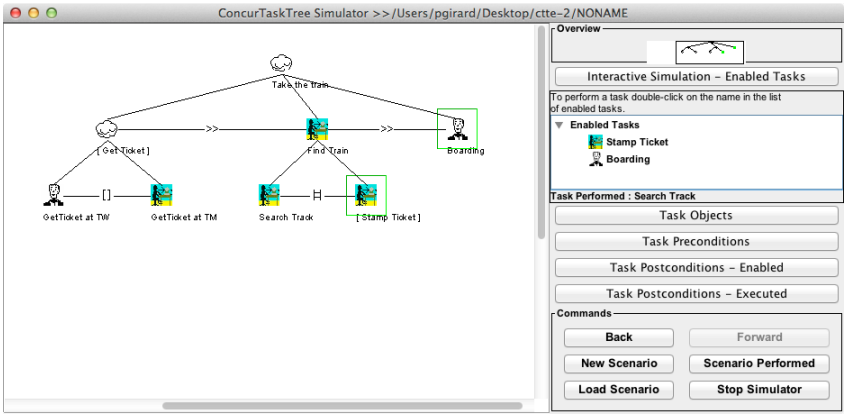


Figure 6: The simulator in CTTE

HAMSTERS also highlights the current enabled tasks in the graphical model, using a colour code to help the user understanding the context: green tasks are enabled, while red tasks refer to skipped tasks. A panel shows the in progress scenario (Figure 7).

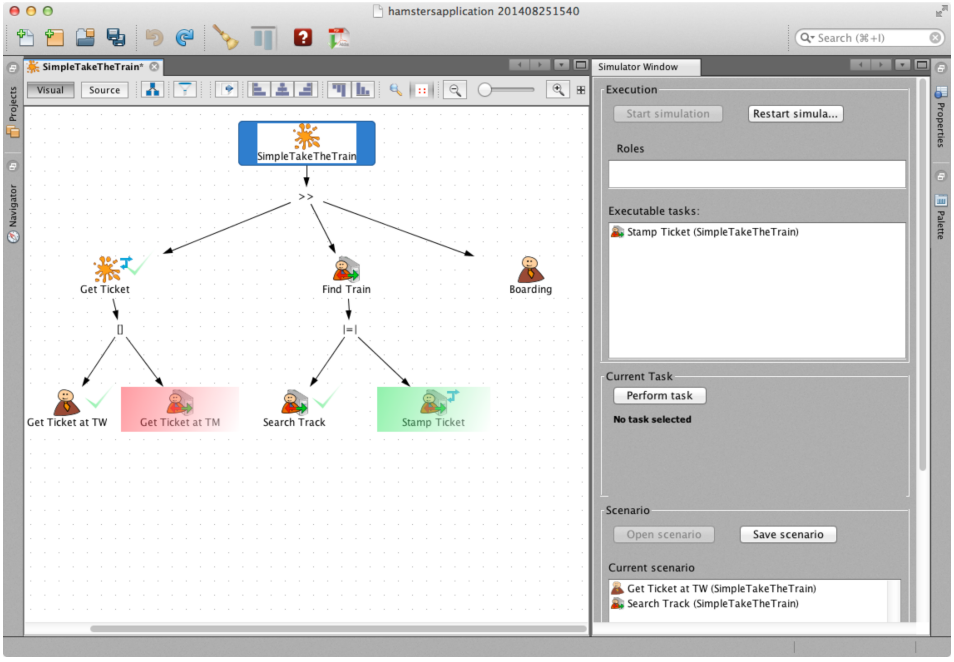


Figure 7: HAMSTERS simulator in action

K-MADe adds information through colour codes about *already done* subtasks (grey), *in progress* tasks (light bue-grey), *passed* tasks (green) and *enabled* tasks (blue). The step in Figure 8 is also during the *Find Train* task, but here, the user decided to pass the *Stamp Ticket* task. Like HAMSTERS, the window includes the scenario in progress. Looking like a debugger, the K-MADe simulator also displays information about the selected task (attributes, and so on), several other information about objects, actors, constraints, etc.

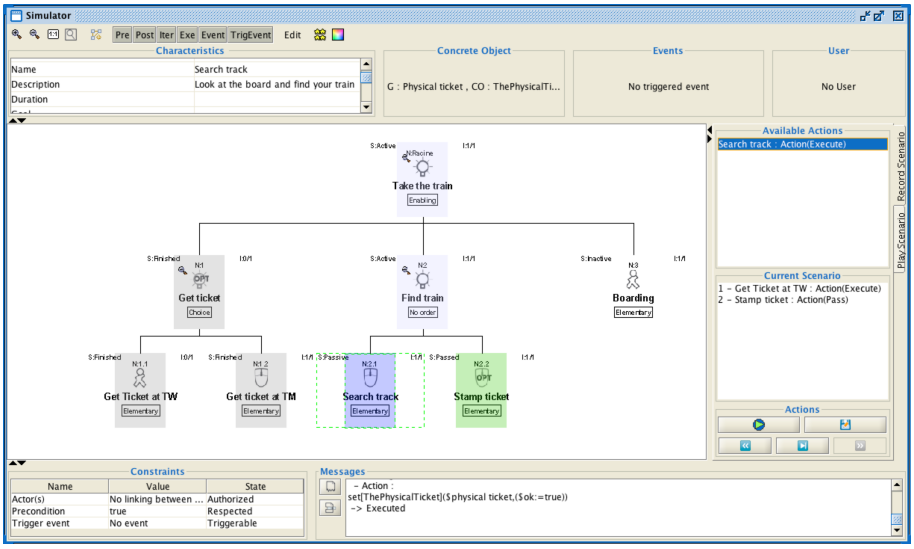


Figure 8: K-MADe simulator in action

The second strategy to help the user understanding task context is proposed by ProtoTask. This tool does not show the tree at all. Instead, it displays relevant information about the current task, and provides a hierarchical view of the scenario, as displayed in Figure 9.

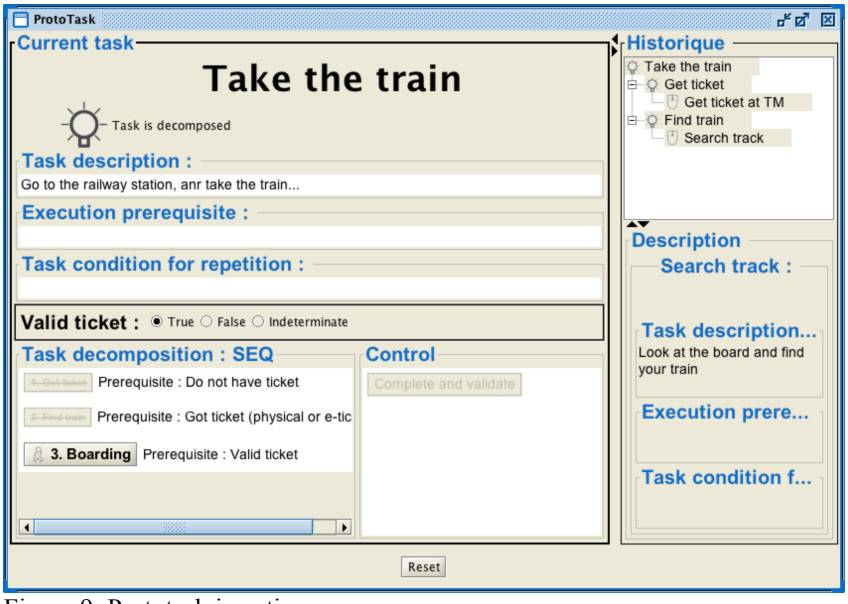


Figure 9: Prototask in action

The main window in Figure 9 focuses on the current task, providing a detailed description (if put into the task model), the different conditions (precondition, iteration condition), and the task decomposition with all enabled tasks. On the top right corner, a navigable scenario is displayed, in a tree-like format, in order to figure out the goal/sub-

goal user strategy. Some experiments showed that this representation is suitable for understanding the global activity [Lachaume, Caffiau, et al. 2012]. In the figure, the step is just after completion of task *Find Train*; the scenario panel shows that the optional *Stamp Ticket* task was not performed.

3.2.3. Summary

Table 1 below summarizes the differences described in this section. Beyond the obvious difference between Prototask and the other three simulators, other differences can be reported. They result from design choices and different policies in the user interface. Note that Prototask is intended to be seen by non-expert users, while the other three systems address mainly task modellers, engineers and domain experts.

LAYOUT	CTTE	HAMSTERS	K-MADe	Prototask
ETS view	simple list of terminal tasks	simple list of terminal tasks	list of terminal tasks with qualification (explicit skip)	list of all sub-tasks and their preconditions
task tree	viewed with colour codes	viewed with colour codes	viewed with colour codes	not visible
scenario	not directly visible during the simulation	visible as a list of terminal tasks during simulation	visible as a list of terminal tasks during simulation	visible as a tree during the simulation
conditions	not directly visible during the simulation	not directly visible during the simulation	not directly visible during the simulation	visible for the current state

Table 1: Summary of differences of visualization

3. 3. BEHAVIOURS

As we described in previous sections, the different simulators lean on the same basic concepts, but differ from each other by several options. The divergences increase when looking at the dynamic behaviour of simulators during simulation. We specifically address four topics. First, we describe the specific case of optional tasks. Then, we focus on task node management and task completion. Next, we address object and pre/post-condition management. After giving an insight on ending model management, we give a summary of all differences between tools.

3.3.1. Optional tasks

The basic principle of task model simulation is to determinate the Enabled Task Sets, i.e. the set of tasks that can be activated at each step of the simulation. This set is calculated according to the semantics of operators. For example, all tasks governed by a ***Choice*** operator are enabled at the same time, while access to tasks governed by a ***Sequential/Enable*** operator is restricted by the order of the sequence, only one task being available at each step. Interactive simulation consists for the user in choosing in the ETS the task to simulate. All simulators work the same way for this point, but differences appear when managing optional attribute and nodes.

In CTTE/HAMSTERS, optional tasks[9] are inserted in the ETS exactly the same way as non-optional tasks. Skipping optional tasks is not a user action. Tasks are skipped because after one step, they are no more in the ETS. For example, Figure 10 shows the "Take the Train" simulation at start.

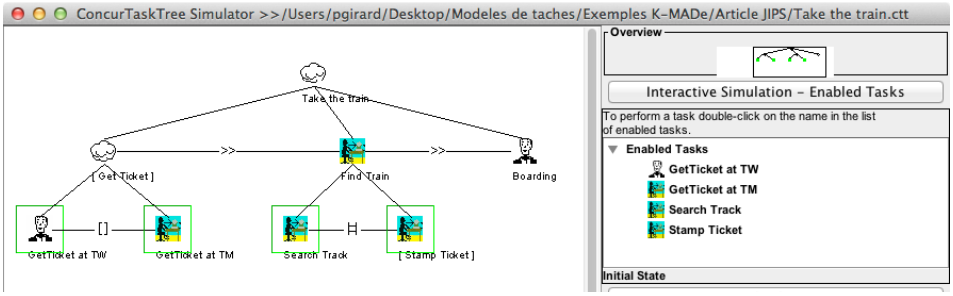


Figure 10: Optional tasks in CTTE

The first two enabled tasks are the terminal tasks of sub-task *Get Ticket*, which is the first task in the first level of sequential decomposition. These two tasks are exclusive (***Choice*** operator). We also can see in the ETS the other two sub-tasks of *Find Train*, which are also enabled. The reason is that *Get Ticket* is an optional task (written in brackets). Implicitly, if the user selects *Search Train* or *Stamp Ticket*, *Get Ticket* is skipped. HAMSTERS and Prototask work the same way.

In K-MADe, optional task management is explicit. The user is requested to explicitly skip the optional task when s/he does not want this task to appear in the current activity (the scenario currently in progress). This can be seen on Figure 11, where the ETS is extracted on the bottom right of the figure. At the start, the user is supposed to get a ticket using one of the two different methods, or to switch directly to the next task. In the last case, s/he must explicitly skip the *Get Ticket* task

[9] We call optional task a task of which the optional attribute is set to true

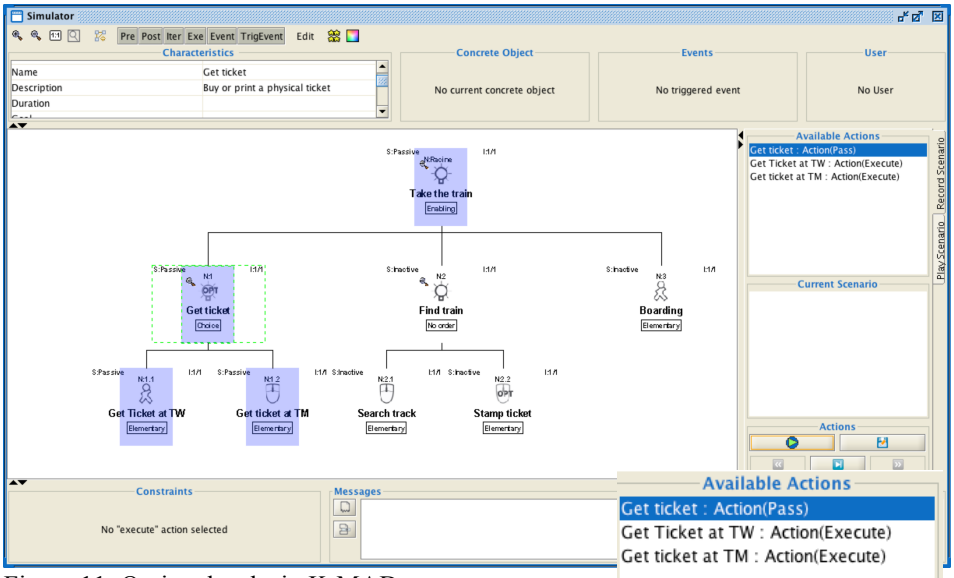


Figure 11: Optional tasks in K-MADe

Figure 12 below shows the situation after skipping the first group of tasks.

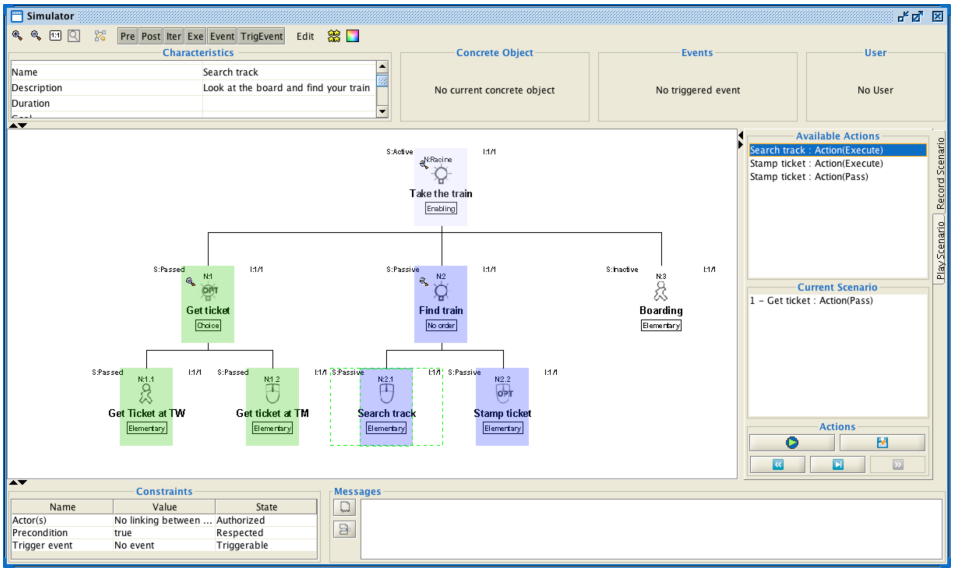


Figure 12: One step further, after skipping the task

3.3.2. Node management and task completion

Enabled Task Sets in CTTE and HAMSTERS only include terminal tasks, i.e. task tree leaves, as introduced in section 3.2. This fact makes it difficult for the user to figure out the whole activity, to know what global task is started, what are the consequences of choosing a subtask, and so on. To solve this problem, tools provide complementary views as described in section 3.2, such as the global tree view. For example, in Figure

13, the context awareness in CTTE is illustrated. Looking at the ETS panel, it is not possible to understand the exact context, which represents the following situation: the *Take the train* task is started, choosing either starts a new subtask, and choosing one of the last two tasks starts a new subtask, *Find train*, making *Get Ticket* unavailable. All this can only be understood through careful analysis of the tree shown on the left. Skipping tasks and ending tasks are all implicit, which make it impossible to have only optional tasks in the ETS.

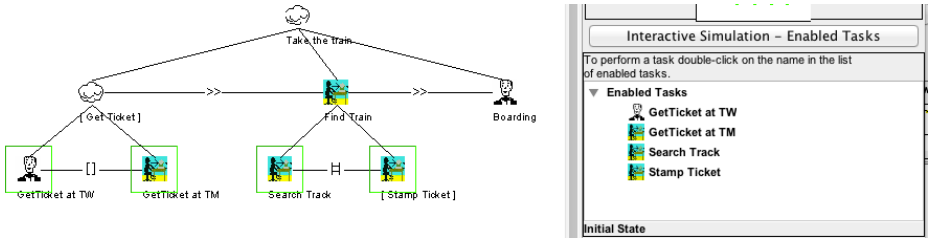


Figure 13: Context awareness in CTTE

K-MADe proposes making the skipping process of all *optional tasks* explicit, what impacts also the node management. Figure 14 shows the K-MADe simulator at the same state as CTTE above. The ETS shows only the first two subtasks, and the option to skip the node task *Get Ticket*. Choosing it enables *Find Train*, but only its two subtasks, because it is not optional itself.

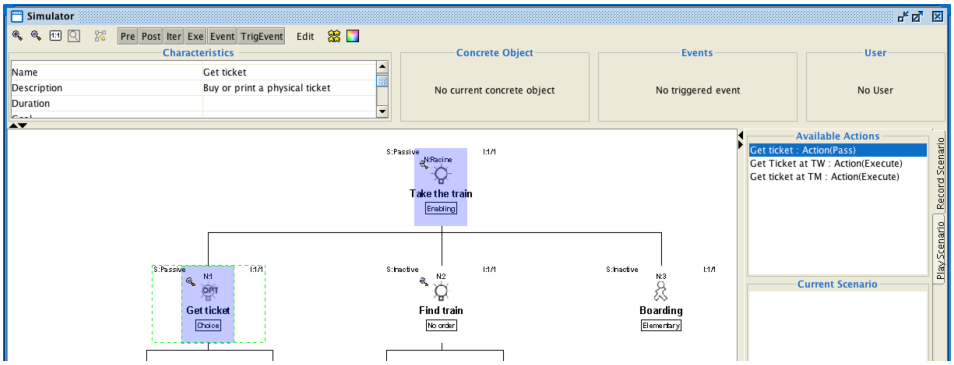


Figure 14: Explicit optional task *skipping*

ProtoTask goes one step further in making explicit the choice of node tasks and the process of ending tasks. Figure 15 shows Prototask after validation of *Get Ticket at TW*; as shown on the top, we are currently in the *Get Ticket* task node, no action is possible with the exception of the final action "***Validate and Terminate"***.

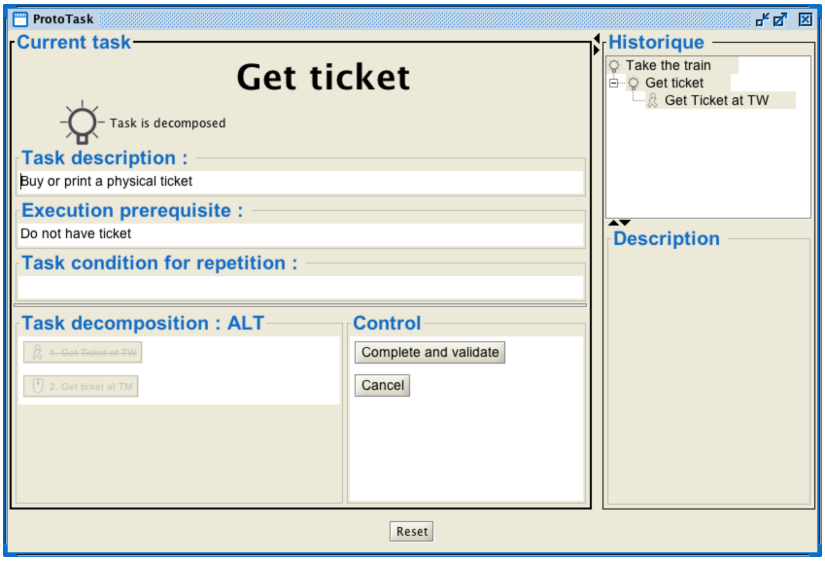


Figure 15: Explicit ending of a task

3.3.3. Object and pre/post-condition management

In recent years, object and pre/post-condition management has become increasingly included in editing tools, and as a consequence, has been better considered in simulators. The increased expressive power given by pre/post-conditions allowed taking into account more realistic activity descriptions. Once again, differences in object management can be observed during simulation. Two different policies are used in the tools. The first one is a human interactive policy, used by CTTE and ProtoTask, and the second one is a process directed policy, used by K-MADe and HAMSTERS.

CTTE and ProtoTask ask the user to personally manage the entire evolution of pre/post-conditions during the simulation. In the control panel of CTTE simulator, several panels allow the exploration and manipulation of objects and pre/post-conditions. Figure 16 shows on the left part the object panel, which gives the description of objects (name, type and value) and their owners (tasks where they are defined), and allows the value to be changed. On the right part, the precondition panel shows the preconditions of every concerned task, with their expression and their current value. We use the word “concerned” to define all possible enabled tasks. The ETS panel only presents enabled tasks, which means tasks that are enabled by the temporal operators and attributes (optional), and whose possible precondition is calculated as true.

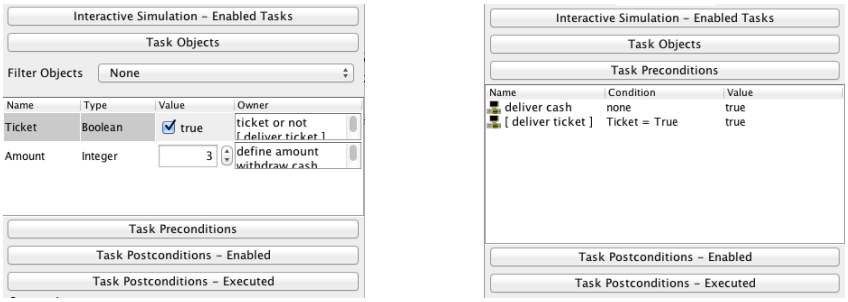


Figure 16: Object (left) and preconditions (right) in CTTE simulator

In the precondition panel, even tasks with false values are shown. In Figure 17, we can see on the left part the only enabled task (*Deliver Cash*), while in the precondition panel (right part), the two potential tasks are displayed, with a "false" value for *Deliver Ticket*.

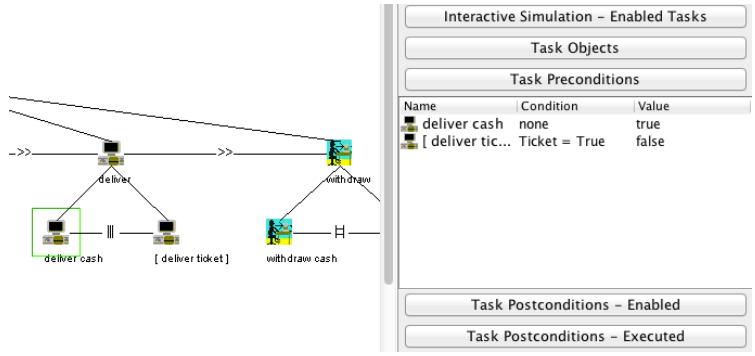


Figure 17: Task with false preconditions in CTTE

In CTTE, using objects and conditions requires the user to manually change the values of objects in order to build scenarios that cover the different options.

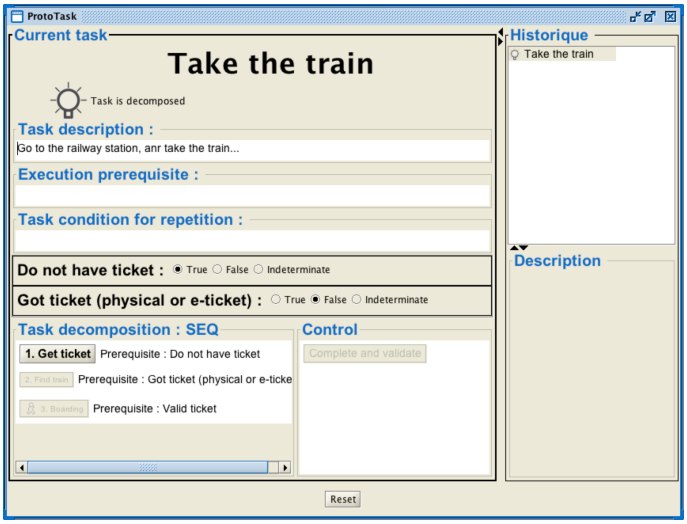


Figure 18: Precondition management in Prototask

ProtoTask uses the same policy, but does not use objects: preconditions are represented by text sentences, which are displayed with True/False/Undetermined status; the user is supposed to give the value s/he wants this precondition to hold for this task in the current scenario. In Figure 18, we can see the situation at the beginning of the "*Take the Train*" activity. In the ETS part, all subtasks of the main task are shown, with their precondition, and only preconditions of concerned tasks are modifiable. Here, the last task (*Boarding*) is not available because the *Search Train* task must be made beforehand.

On the opposite, K-MADe and HAMSTERS do not really allow an interactive modification of object values during simulation. Instead, they provide a way to describe side effects on objects. Tasks are able to define assignations (HAMSTERS) or actions (K-MAD), which allow user to change the values of objects. For example, in K-MADe, *Get Ticket at TW* is supposed to set a *Physical Ticket* boolean value to true. This is done through an action, which is visible on Figure 19a when selecting this action in the ETS panel, while after execution of this task, the feedback messages explain that the action is done (Figure 19b).

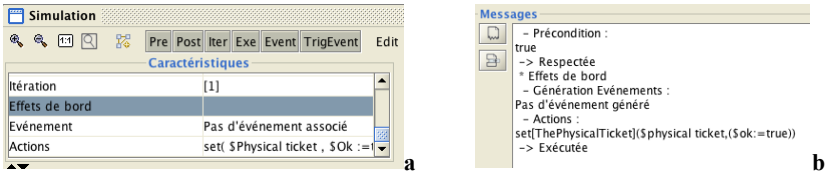


Figure 19: Action management in K-MADe

Thanks to this management, preconditions can be correctly evaluated during the complete simulation.

Note that no simulator manages correctly the precondition and optional attribute association. Optionality is always a user choice. So, it is not possible to state that a task is optional depending on a precondition, i.e. mandatory if the precondition is true, and systematically omitted if the condition is false.

The last difference in managing pre/post-conditions is in the way simulators give access to tasks whose precondition is false. In CTT and HAMSTERS, these tasks are considered as not enabled, which means they are not in the ETS view. In ProtoTask, they are greyed, the same way not enabled tasks are. In K-MADe, such tasks appear in the ETS panel exactly as if they were enabled; if the user tries to simulate them, an error message gives a feedback to the user, explaining that the precondition is not fulfilled.

3.3.4. Managing end of simulation

The last main behavioural difference for common features in simulators can be seen in managing the end of simulations. What does "ending" the activity really mean? Using only temporal operators, the answer is quite simple: when no more tasks are present in the Enabled Task Set, the activity can be considered as completed.

Nevertheless, when adding attributes and preconditions, the problem becomes more difficult. If the last task in the ETS is optional, how should the fact that it is not performed be represented? If the only possible task is guarded by a precondition, and this condition is false, what does it means?

In CTT and HAMSTERS, only enabled tasks, with preconditions set at true, are available in the ETS. So, when the ETS is empty, the activity is supposed to be

completed. There is no way to state that the task is completed even if an optional task is always available. To eliminate this problem of optional tasks, a rule in CTT states that it is not possible to end a model with optional tasks. Nevertheless, the CTTE simulator does not manage this case.

In K-MADe, a specific case for optional tasks was defined. When an optional task is enabled, the user is asked to run it, or to skip it. This is an explicit choice for stating that an optional task is not to be performed.

In ProtoTask, the problem is solved by the specific notion of task completion. The user is asked to assert that a task is completed. So, when a task is optional, the user can choose between this task and the explicit ending of the node. For example in Figure 20, the *Stamp Ticket* task is optional, so the ProtoTask's user can choose between running the task or completing (***Validate and Complete*** button) the *Find Train* task.

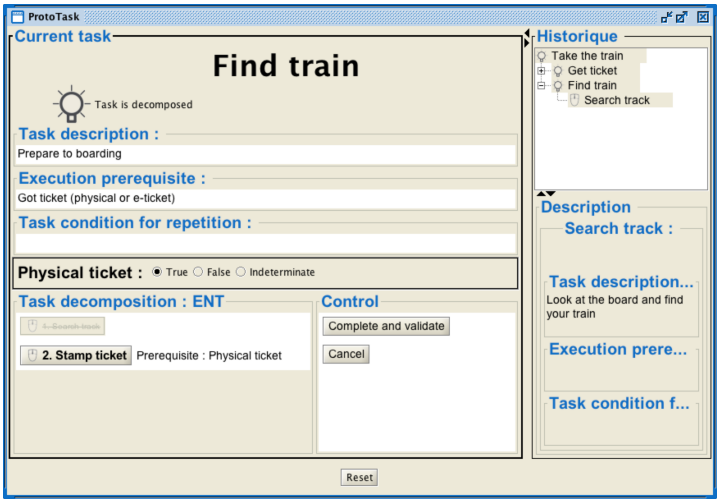


Figure 20: Task completion in ProtoTask

3.3.5. Summary of behavior differences

Table 2 summarizes the differences described in this section. We can see that these differences do not determine two categories of distinct simulators. For example some of them converge for simulators which have very different objectives, such as CTTE and Prototask for the condition management policy.

BEHAVIOUR	CTTE	HAMSTERS	K-MADe	Prototask
optional tasks	implicit skip	implicit skip	explicit skip	explicit skip
node management	none	none	explicit management of optional nodes	explicit management
task completion	implicit	implicit	implicit	explicit control
object/conditions management policy	human interactive	process driven	process driven	human interactive
objects	different predefined object types	Booleans	user-built and predefined object types	no objects
pre/post-condition	pre-defined Boolean expressions	Boolean expressions	free Boolean expressions	textual expressions
side-effects on objects	no	yes (assignations)	yes (actions)	no
control of guarded tasks	proactive (not visible)	proactive (not visible)	post-control (appears enabled)	proactive (appears disabled)
Managing end of simulation	automatic	automatic	controlled for optional ending tasks	controlled

Table 2: Summary of behaviour differences

4. SIMULATOR USAGE

Within the task modelling literature, no comprehensive reviews of the different simulators have yet been compiled. Whilst they appear in some tool descriptions, or in description of case studies, the only article where simulators were compared in terms of usage is [Paternò 2002], which only includes two simulators, CTTE and VTMB (no more available for use by practitioners). Nevertheless, by extracting details from different articles, it is possible to state what kind of simulator's usage has been made. Three main kinds of usage may be noticed: understanding, exploring and explaining, and validating.

4.1. UNDERSTANDING

The first usage of simulators results form the fact that simulators enforce and make explicit the semantics of the task modelling notations: it allows users to understand with no doubt the actual dynamic semantics of the model they are looking at. As we saw in previous sections, some behaviours are not really described in method presentations, and using the simulator is the only way to know what interpretation must be made.

This point makes the choice to use simulators while teaching task modelling obvious. As reported in [Caffiau, Scapin, et al. 2008], using simulation is very important for students to figure out what the task model really models. The coordinated views of the ETS and the current task model are particularly suitable to understand the context of the current state, and to master the different elements of the notation. CTTE, for example, changes the task model display after each task simulation, centring the task

model on the current enabled tasks. Displaying the scenario which is in progress is provided by K-MADe and HAMSTERS, helping users to remember what they did. K-MADe provides a very complete view of the task model, which allows to explore every element of the model. Prototask does not show the task model itself during the simulation. Instead, it makes explicit the goal/subgoal hierarchy, by asking the user to enter a subtask, and by always showing the current scenario, in a tree-like representation. In [Lachaume, Girard, et al. 2012], the authors provide results that proved a high level of user understanding of the task model.

4.2. EXPLORING AND EXPLAINING

The second usage originates from the first goal of simulators: animating the dynamic semantics of models. The result is a visual and dynamic representation of this dynamics, which makes clear the behaviour that underlies the activity.

As pointed out in [Paternò 2002], the simulation may then be the support for a multidisciplinary discussion between people with different background to take design decision at the task level, during early stages of system design. Nevertheless, understanding what happens during simulation requires a minimum knowledge about task modelling; indeed, people involved in this kind of activity must be able to interpret complex tree views, with coded representations. Reported experiments [Paternò et al. 2012; Martinie et al. 2012] confirmed the correctness of this approach.

On the opposite, using traditional task model simulators for interacting with end-users or stakeholders seems more difficult, as established in [Caffiau, Guittet, et al. 2008]. The ProtoTask approach [Lachaume, Girard, et al. 2012], where the task tree is never shown to the user, proved to be efficient in allowing end-users to validate their needs. Improved visualisation of in progress scenarios and explicit usage of task/subtasks decompositions allow end-users to correctly figure out the simulated activity.

4.3. VALIDATING

Validating modelled activities is a crucial challenge. The first need is validating scenarios. Simulators allow users to build scenarios while simulating models. Recorded scenarios may then be checked over the actual systems, or played again on the simulator after changes [Girard et al. 2013]. Nevertheless, this topic has not been yet fully investigated, and we propose new directions in the conclusion.

On the opposite, linking task modelling simulators to system models has been investigated in [Barboni et al. 2010]. The HAMSTERS environment is linked to the PetShop environment [Bastide et al. 2002] to allow co-execution of task and system models. This way, designers are able to validate the actual behaviour of the system compared to task models.

5. CONCLUSION AND PERSPECTIVES

In this article, we propose a comparison of currently available and maintained task model simulators. This review highlights differences in behaviour, which are not necessarily explained in the tool presentations.

Despite the wide range of usages described in section 4, several challenges remain for task model simulators. They can be split in two categories: modelling challenges, and usage challenges.

Task methods incorporate more and more concepts, increasing greatly their expressive power. In the meantime, simulators do not take into account some of them. A first example is parallelism. While parallel operators are generally available in the notations, managing parallelism during the simulation is not really done, and feedback of parallel tasks is very poor. A second example relates to interruptions. Interrupting, and possibly resuming a task is a very complex process, which relates to very different things, such as cognitive aspects for the user, interrupting and resuming processes, and so on. Time is also a topic that is not really taken into account during simulation. Verification of timing issues can be made during simulation, but no tool seems to manage it yet.

In section 3, we explored how the different simulators manage objects and pre/post-conditions. This management is very different from one tool to others. What should really be the place of objects and pre/post-conditions in task models? What should be their role in the simulation? All these points need more investigation. Problems remain in using simulators. The major example relates to scenarios. Current scenarios record only tasks. They do not consider the context of the tasks. Replaying scenarios requires to take care of this context; how is it possible to obtain the same result if the conditions of tasks are different? No simulator takes care of this point at the moment. In the same way, exploring models and modifying parameters during simulation is not really standardised over the different simulators.

Despite academic attempts, task models are not widely used in system design processes. Connections between task models and UML models, or process modelling notation such as BPMN[10] for example, are not really clear. The already mentioned W3C initiative for standardizing task models did not lead to a significant usage of task models in current software engineering methods. Agile methods promote extensive discussion between users (or product owners) and developers using simple descriptions. Despite of the reported experiences with CTTE or Prototask, which seems to demonstrate that this dialogue can be improved using task-modelling tools, task models have not yet found their place in this process.

The last point concerns simulator usability. Many differences between the different tools result from strong design choices and tool policy (as for example explicitly skipping optional tasks, or explicitly activating task nodes). Authors demonstrated the correctness of their choices, but no comparative study has been made between these different tools. More, whereas task modelling comes from ergonomists and psychologists, it seems that advanced tools, and more precisely those with simulators, are mainly used by software engineers and HCI designers. The reason of this situation should be investigated, and comparative studies could bring interesting points about task model acceptance by different publics and task analysis and modelling benefits.

6. BIBLIOGRAPHY

ANNETT, J., 2004. Hierarchical Task Analysis . In D. Diaper & N. Stanton, eds. *The handbook of task analysis for Human-Computer Interaction*. Lawrence Erlbaum Associates, pp. 67–82.

ANNETT, J., DUNCAN, K.D., STAMMERS, R.B. AND GRAY, M.J., 1971. *Task analysis.*, London: Her Majesty's Stationery Office.

[10] http://www.bpmn.org

BARBONI, E., LADRY, J.-F., NAVARRE, D., PALANQUE, P. AND WINCKLER, M., 2010. Beyond modelling: An Integrated Environment Supporting Co-Execution of Tasks and Systems Models. In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '10*. New York, New York, USA: ACM Press, p. 165.

BARON, M., LUCQUIAUD, V., AUTARD, D. AND SCAPIN, D., 2006. K-MADe : un environnement pour le noyau du modèle de description de l'activité. In J.-M. Robert & B. David, eds. *IHM'06*. Montréal, Canada: ACM Publishers, pp. 287–288.

BASTIDE, R., NAVARRE, D. AND PALANQUE, P., 2002. A model-based tool for interactive prototyping of highly interactive applications. In *CHI '02 extended abstracts on Human factors in computing systems - CHI '02*. New York, New York, USA: ACM Press, pp. 516–517.

BAUMEISTER, L.K., JOHN, B.E. AND BYRNE, M.D., 2000. A comparison of tools for building GOMS models. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*. New York, New York, USA: ACM Press, pp. 502–509.

BIERE, M., BOMSDORF, B. AND SZWILLUS, G., 1999. The Visual Task Model Builder. In J. Vanderdonkt & A. Puerta, eds. *Third Conference on Computer-Aided Design of User Interfaces (CADUI'99)*. Louvain-la-neuve, Belgique: Kluwer Academic Publishers, pp. 245–256.

BIERE, M., BOMSDORF, B. AND SZWILLUS, G., 1999. The Visual Task Model Builder. In *Proceedings of the third international conference on Computer-aided design of user interfaces*. Norwell, MA, USA: Kluwer Academic Publishers, pp. 245–256.

CAFFIAU, S., GUITTET, L., SCAPIN, D.L. AND SANOU, L., 2008. Utiliser les outils de simulation des modèles de tâches pour la validation des besoins utilisateur : une revue des problèmes. In *ERGO'IA*. Biarritz, France, pp. 257–258.

CAFFIAU, S., SCAPIN, D.L. AND SANOU, L., 2008. Retour d'Expérience en Enseignement de la Modélisation de Tâches. In *ERGO'IA*. Biarritz, pp. 135–143.

CARD, S., MORAN, T. AND NEWELL, A., 1983. *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates.

DELOUIS, I. AND PIERRET, C., 1991. Emad : Manuel de référence .

DIAPER, D., 1989. Task Analysis for Knowledge Descriptions (TAKD): The method and an example. In D. Diaper, ed. *Task analysis for human- computer interaction*. Chichester, UK: Ellis Horwood, pp. 108–159.

DIAPER, D., 2004. Understanding Task Analysis for Human-Computer Interaction. In D. Diaper & N. Stanton, eds. *The Handbook of Task Analysis for Human-Computer Interaction*. Mahwah: Lawrence Erlbaum Associates, Inc., pp. 5–48.

GAMBOA, R.F., SCAPIN, D.L., HANSMANN, W., HEWITT, W.T. AND PURGATHOFER, W., 1997. Editing MAD* task description for specifying user interfaces, at both semantic and presentation levels. In M. D. Harrison & J. C. Torres, eds. *Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'97)*. Granada, Spain: Springer-Verlag, pp. 193–208.

GIESE, M., MISTRZYK, T., PFAU, A., SZWILLUS, G. AND DETTEN, M. VON, 2008. AMBOSS: A Task Modeling Approach for Safety-Critical Systems. In P. Forbrig & F. Paternò, eds. *Engineering Interactive Systems (HCSE 2008 and TAMODIA 2008)*. Pisa, Italy: Springer (LNCS 5247), pp. 98–109.

GIRARD, S. ET AL., 2013. *Artificial Intelligence in Education* H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik, eds., Berlin, Heidelberg: Springer Berlin Heidelberg.

HIX, D. AND HARTSON, H.R., 1993. *Developping user interfaces: Ensuring usability through product & process*, Newyork, USA: John Wiley & Sons, inc.

JOHN, B.E. AND KIERAS, D.E., 1996. The GOMS Family of User Interface Analysis Techniques: Comparaison and Contrast. *ACM Transactions on Computer-Human Interaction*, 3(4), pp.320–351.

JOHNSON, H. AND JOHNSON, P., 1991. Task Knowledge Structures: Psychological basis and integration into system design. *Acta Psychologica*, 78, p.3-26.

JOHNSON, P., JOHNSON, H., WADDINGTON, R. AND SHOULS, A., 1988. Task-related knowledge structures: analysis, modelling and application. In *Proceedings of the Fourth Conference of the British Computer Society on People and computers IV*. Cambridge University Press, pp. 35–62.

JOHNSON, P., WILSON, S., MARKOPOULOS, P. AND PYCOCK, J., 1993. ADEPT: Advanced Design Environment for Prototyping with Task Models. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '93*. New York, New York, USA: ACM Press, p. 56.

JOURDE, F., LAURILLAU, Y. AND NIGAY, L., 2010a. COMM notation for specifying collaborative and multimodal interactive systems. In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '10*. New York, New York, USA: ACM Press, pp. 125–134.

JOURDE, F., LAURILLAU, Y. AND NIGAY, L., 2010b. e-COMM, un éditeur pour spécifier l'interaction multimodale et multiutilisateur. In *Conference Internationale Francophone sur I'Interaction Homme-Machine on - IHM '10*. New York, New York, USA: ACM Press, pp. 225–228.

KIERAS, D.E., 2004. GOMS Models for Task Analysis. In D. Diaper & N. Stanton, eds. *The handbook of Task Analysis*. pp. 83–116.

LACHAUME, T., CAFFIAU, S., GIRARD, P., FOUSSE, A. AND GUITTET, L., 2012. Comparaison de différentes approches de simulation dans les modèles de tâches. In *Ergo-IHM'2012*. Biarritz, France: ACM, pp. 60–67.

LACHAUME, T., GIRARD, P., GUITTET, L. AND FOUSSE, A., 2012. ProtoTask, new task model simulator. In M. Winckler, P. Forbrig, & R. Bernhaupt, eds. *Human-Centered Software Engineering*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 323–330.

LIMBOURG, Q., VANDERDONCKT, J., MICHOTTE, B., BOUILLON, L. AND LOPEZ-JAQUERO, V., 2005. UsiXML: a Language Supporting Multi-Path Development of User Interfaces | UsiXML - USer Interface eXtended Markup Language. In R. Bastide, P. Palanque, & J. Roth, eds. *Engineering Human Computer Interaction and Interactive Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 200–220.

MARTINIE, C., 2011. *Une approche à base de modèles synergiques pour la prise en compte simultanée de l'utilisabilité, la fiabilité et l'opérabilité des systèmes interactifs critiques*. Toulouse, France.

MARTINIE, C., PALANQUE, P., NAVARRE, D. AND BARBONI, E., 2012. A development process for usable large scale interactive critical systems: application to satellite ground segments. In M. Winckler, P. Forbrig, & R. Bernhaupt, eds. *HCSE'12 Proceedings of the 4th international conference on Human-Centered Software Engineering*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 72–93.

MORI, G., PATERNO, F. AND SANTORO, C., 2002. CTTE: support for developing and analyzing task models for interactive system design. *IEEE Transactions on Software Engineering*, 28(8), pp.797–813.

PATERNO, F., 1999. *Model-Based Design and Evaluation of Interactive Applications*, Springer.

PATERNO, F., 2002. Tools for Task Modelling: Where we are, Where we are headed. In *TAMODIA '02 Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design*. INFOREC Publishing House Bucharest, pp. 10–17.

PATERNO, F., FACONTI, G.P. AND COMPUTER, P. AND, 1992. On the LOTOS use to describe graphical interaction. In Cambridge University Press, pp. 155–173.

PATERNO, F., MANCINI, C. AND MENICONI, S., 1997. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *IFIP TC13 human-computer interaction conference (INTERACT'97)*. Sydney, Australia, pp. 362–369.

PATERNO, F., MORI, G. AND GALIMBERTI, R., 2001. CTTE: An Environment for Analysis and Development of Task Models of Cooperative Applications. In *ACM CHI 2001*. Seattle: ACM Press.

PATERNO, F., SANTORO, C. AND SPANO, L.D., 2012. Improving support for visual task modelling. In M. Winckler, P. Forbrig, & R. Bernhaupt, eds. *HCSE'12 Proceedings of the 4th international conference on Human-Centered Software Engineering*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 299–306.

SCAPIN, D.L. AND PIERRET-GOLBREICH, C., 1990. Towards a method for task description : MAD. In L. Berliguet & D. Berthelette, eds. *Working with Uisplay units*. Elsevier Science Publishers, North-Holland, pp. 371–380.

SEBILLOTE, S., 1992. Task analysis and formalization according to MAD: Hierarchical task analysis, method of data gathering and examples of task description.

SEBILLOTTE, S. AND SCAPIN, D.L., 1994. From Users' Task Knowledge to High-Level Interface Specification. *International Journal of Human-Computer Interaction*, 6(1), pp.1–15.

STUART, J. AND PENN, R., 2004. TaskArchitect: taking the work out of task analysis. In *Proceedings of the 3rd annual conference on Task models and diagrams - TAMODIA '04*. New York, New York, USA: ACM Press, p. 145_154.

SYSTEMS, I.S.O.I.P., 1984. Definition of the Temporal Ordering Specification Language LOTOS.

TABARY, D. AND ABED, M., 2002. A software environment task object-oriented design (ETOOD). *Journal of Systems and Software*, 60(2), pp.129–140.

TABARY, D., ABED, M. AND KOLSKI, C., 2000. Object-oriented modelling of manual, automatic, interactive task in mono or multi-user contexts using the TOOD method. In Z. Binder, ed. *IFAC 2nd Conference on Management and Control of Production and Logistics (MCPL 2000)*. Amsterdam: Elsevier Science Publisher, pp. 101–111.

TARBY, J.-C. AND BARTHET, M.-F., 1996. The Diane+ method. In J. Vanderdonckt, ed. *Proceedings of the Second International Workshop on Computer-aided Design of User Interfaces (CADUI '96)*. Namur, Belgique: Presses Universitaires de Namur, pp. 95–119.

VAN DER VEER, G.C., 1996. GTA: Groupware Task Analysis - Modeling Complexity. *Acta Psychologica*, 91, pp.297–322.

VAN WELIE, M., VAN DER VEER, G.C. AND ELIËNS, A., 1998. Euterpe - Tool support for analyzing cooperative environments. In *Ninth European Conference on Cognitive Ergonomics*. Limerick, Ireland.

WURDEL, M., SINNIG, D. AND FORBRIG, P., 2008. CTML: Domain and Task Modeling for Collaborative Environments. *Journal of Universal Computer Science*, 14(19), pp.3188–3201.

Thomas Lachaume is preparing his PhD in Computer Science, in the Data Engineering and Human Computer Interaction team of the Laboratory of Computer Science and Automatic Control for Systems (LIAS). He is working more specifically on task models, and task simulators. He developed ProtoTask, a new paradigm of task simulation for task models.

Laurent Guittet is Assistant Professor at the ISAE-ENSMA engineering school, located in Poitiers, France. He belongs to the Data Engineering and Human Computer Interaction team of the LIAS Laboratory. He has been working on software engineering aspects of Human Computer Interaction, and more specifically on architecture models for HCI, HCI for teaching, and task models

Patrick Girard is Professor at the University of Poitiers, France, and head of the HCI group of the Data Engineering and Human Computer Interaction team of the Laboratory of Computer Science and Automatic Control for Systems (LIAS). His main research interests focus on End-User Programming and Task Modelling.

Allan Fousse is Assistant Professor at the University of Poitiers, France, and belongs to the Data Engineering and Human Computer Interaction team of the Laboratory of Computer Science and Automatic Control for Systems (LIAS). Coming from Software Engineering and Data Imaging, he is working now on task modelling and post-WIMP interfaces.