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 .

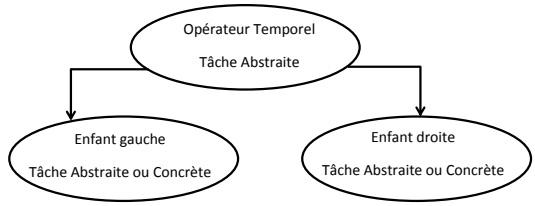


Fig. 1. Exemple de décomposition de tâches

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.

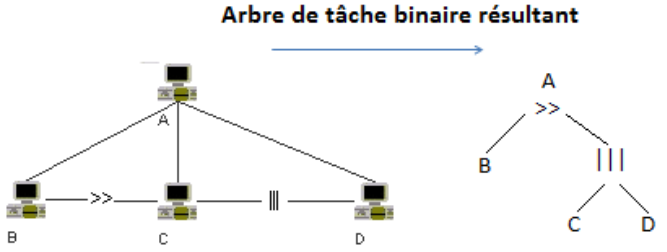


Fig. 2. Arbre de tâches binaire résultant de la transformation

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.

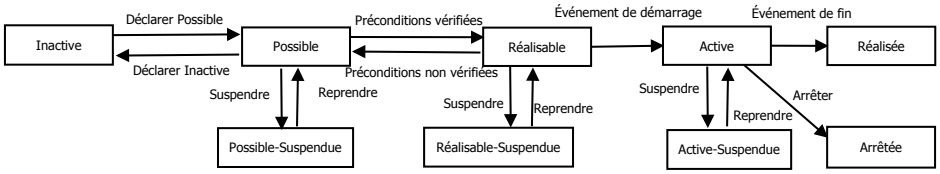


Fig. 3. Les états possibles d'une tâche sous forme d'une machine à état finis

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

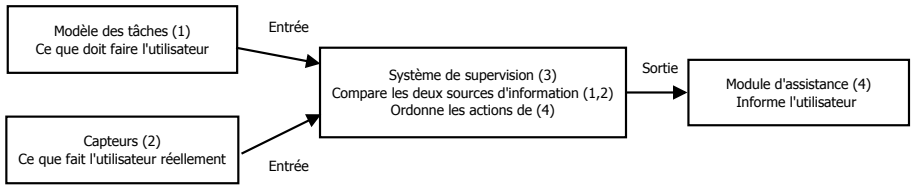


Fig. 4. Approche globale

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.

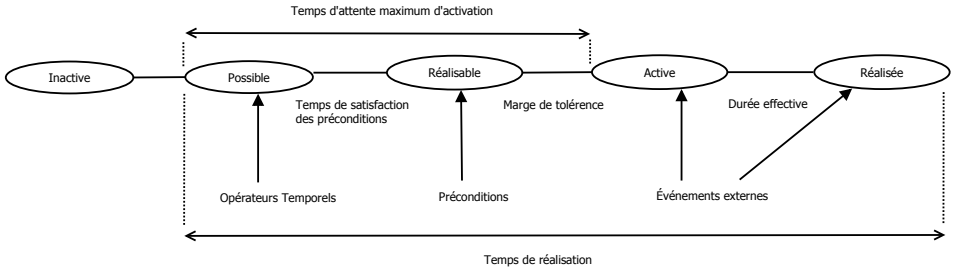


Fig. 5. Cycle de vie nominal d'une tâche

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

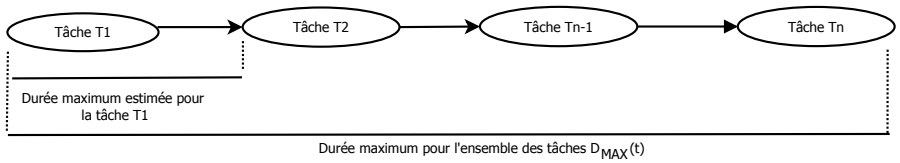


Fig. 6. Temps maximum attribué à un ensemble de tâches

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

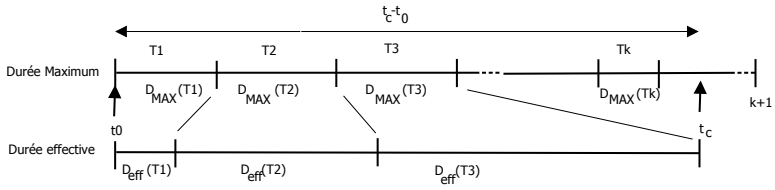


Fig. 7. Temps maximum et effectifs d'un ensemble de tâches

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).

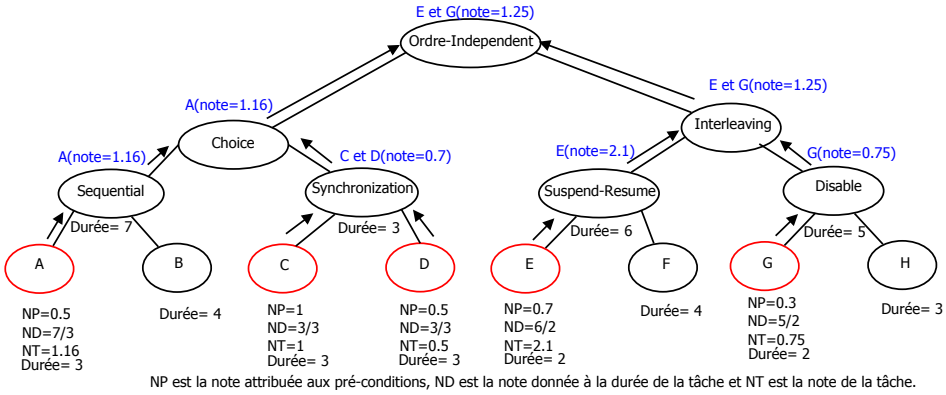


Fig. 8. Exemple de déroulement du processus d'élection d'une tâche candidate

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.

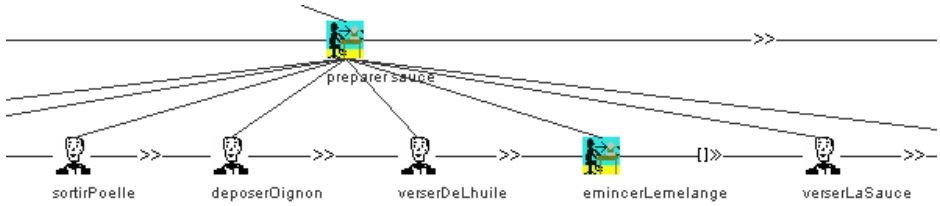


Fig. 9. Exemple d'une partie d'un arbre saisi avec CTTE

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.

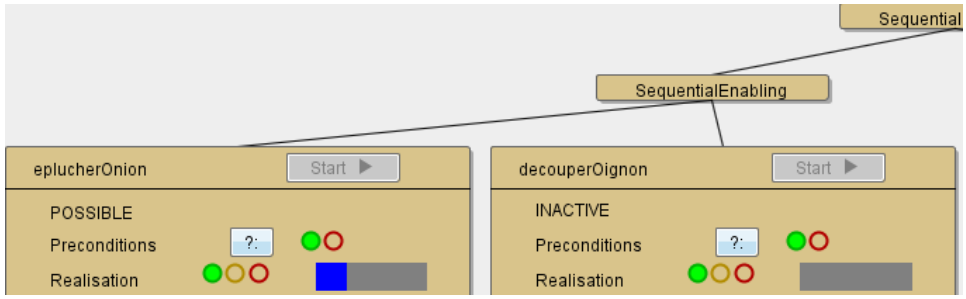


Fig. 10. Notre arbre de tâches binaire

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.

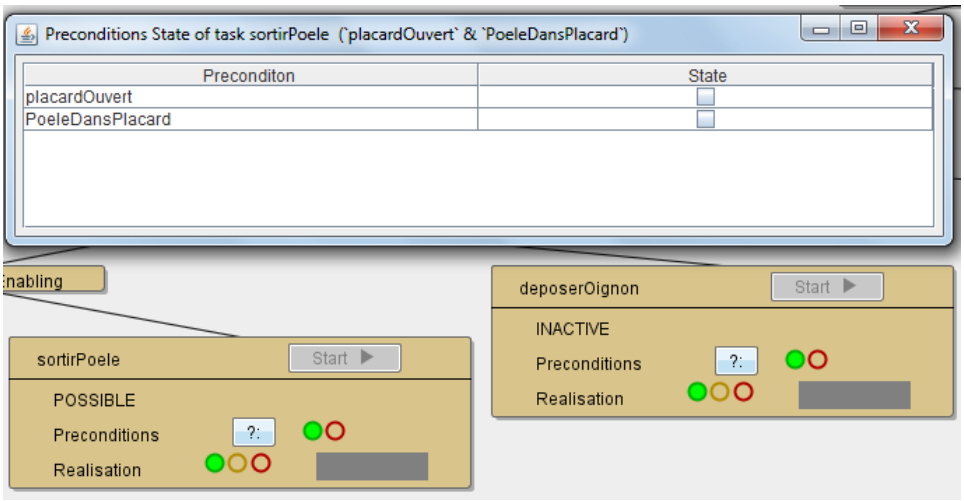


Fig. 11. Représentation des préconditions d'une tâche

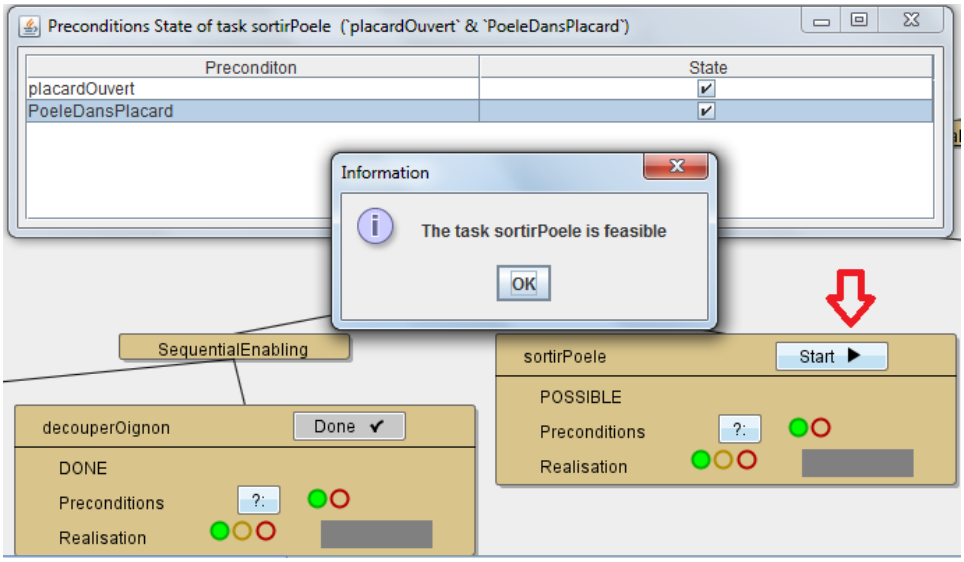


Fig. 12. Vérification des pré-conditions d'une tâche qui devient faisable

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

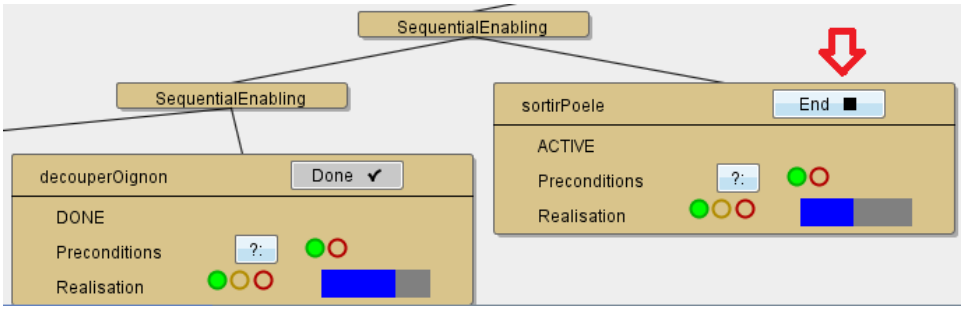


Fig. 13. Une tâche active peut être terminée

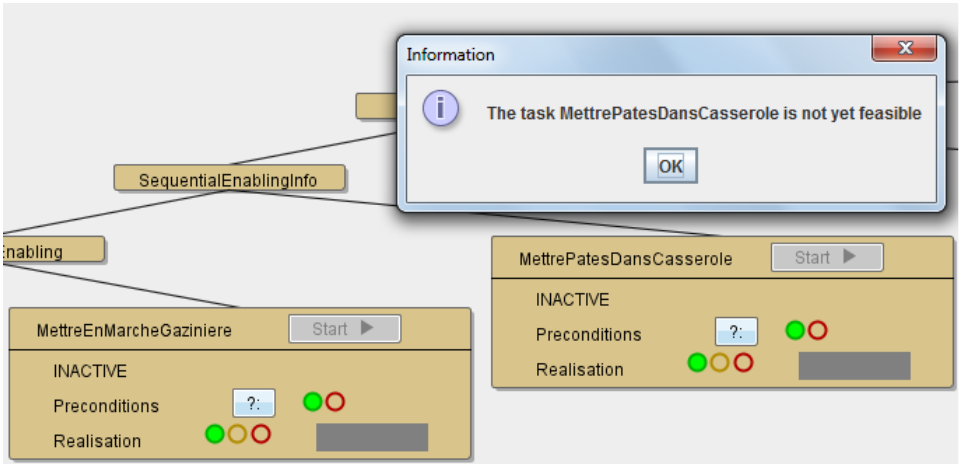


Fig. 14. le CTC notifie une tâche non faisable voulant démarrer

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.

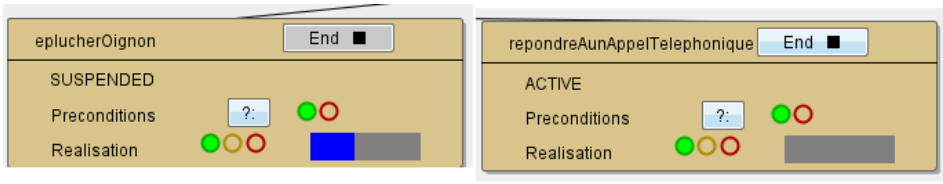


Fig. 15. L'activité en cours suspendue par un appel

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

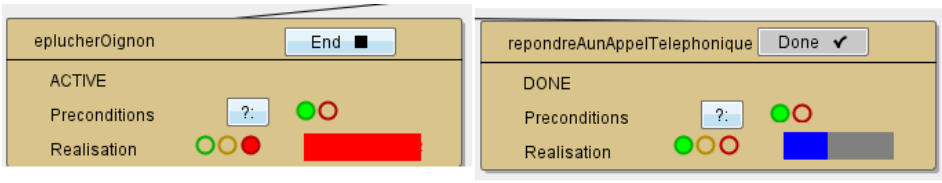


Fig. 16. Reprise de la tâche suspendue suite à la fin de l'appel et notification par le CTA du retard engendré

(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.

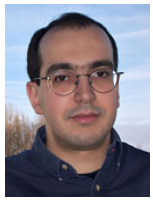
VANDERVEER, G. C., VANDERLENTING, B. F., AND BERGEVOET, B. A. J. 1996. Gta : Groupware task analysis - modeling complexity. *Acta Psychologica 91*, 297–322.

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.