Un système d'aide et de suivi des tâches utilisateur dans un environnement ambiant

Asma Gharsellaoui, Bellik Yacine, Christophe Jacquet

Un système d'aide et de suivi des tâches utilisateur dans un environnement ambiant

Asma Gharsellaoui
LIMSI-CNRS, Université Paris Sud
91403 Orsay Cedex, France
asma.gharsellaoui@limsi.fr

Yacine Bellik
LIMSI-CNRS, Université Paris Sud
91403 Orsay Cedex, France
yacine.bellik@limsi.fr

Christophe Jacquet
Dép.informatique-Supélec
91192 Gif-sur-Yvette Cedex, France
christophe.jacquet@supelec.fr

RÉSUMÉ
Les modèles de tâches existants sont généralement statiques (non utilisés au moment de l'exécution) et exploités uniquement pour la conception ou l'évaluation prédictive des systèmes interactifs. Nous proposons d'utiliser le modèle de tâches au moment de l'exécution, afin de suivre les actions de l'utilisateur, vérifier qu'il n'a pas fait d'erreurs et lui procurer de l'aide en cas de besoin. Nous présentons un modèle de tâches adapté aux environnements ambiants qui attribue dynamiquement des états aux tâches au moment de l'exécution. Nous décrivons également le fonctionnement d'un système de suivi et d'assistance qui exploite notre modèle de tâches dynamique. Nous présentons ensuite une validation de notre système à travers une simulation qui montre comment les interactions avec le modèle de tâches à l'exécution 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.

Mots Clés
Modélisation des tâches utilisateurs ; intelligence ambiante ; interaction dans les environnements ambiants.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces.

INTRODUCTION

Le terme intelligence ambiante (AmI) a été introduit initialement par la Commission Européenne en 2001 [6], [25], bien que le concept en lui-même remonte au début des années 90 [34] [35]. Il désigne une vision de l'environnement quotidien de l'utilisateur dans lequel la technologie est omniprésente et discrète [24]. Un système d'intelligence ambiante est un système interactif dans lequel des objets intelligents embarqués dotés de capacités de calcul et de communication coopèrent afin de répondre au mieux aux besoins des utilisateurs humains [26] [11]. Les systèmes ambiants ont pour objectif de percevoir et d'agir sur l'environnement physique, de façon à s'intégrer dans les activités quotidiennes des utilisateurs. 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...), nous soutenons que le système doit être capable de suivre le déroulement des tâches utilisateur et de l'assister en cas de besoin. Dans la première section de cet article, nous nous intéressons aux spécificités de la modélisation des tâches dans les environnements ambiants. La deuxième section présente le modèle de tâches que nous proposons, et qui est exploitable à l'exécution. Le système d'aide et de suivi est ensuite présenté en détaillant ses stratégies d'intervention. Dans une dernière partie, nous décrivons le simulateur qui a été développé et nous détaillons son fonctionnement à travers le déroulement d'un scénario dans les environnements ambiants.

http://dx.doi.org/10.1145/2670444.2670451

LA MODÉLISATION DES TÂCHES DANS LES ENVIRONNEMENTS AMBIANTS

Besoins de la modélisation des tâches dans les environnements ambiants

Un modèle de tâches décrit les actions destinées à être effectuées afin d'atteindre les objectifs de l'utilisateur [27] et les différentes manières de les atteindre [18]. La modélisation des tâches vise en particulier à construire un modèle qui décrit précisément les relations entre les différentes tâches [21]. Ces modèles sont employés pour des finalités variées telles que l'aide à la compréhension de l'activité, la conception de programmes de formation [2] [3] [1] [22], la conception et l'évaluation des interfaces homme-machine [4] [13]. À cet effet, plusieurs catégories d'utilisateur sont amenés à produire ces modélisations, des utilisateurs novices (par exemple les étudiants lors des formations) aux experts. Plusieurs recherches [17] [19] [23] se sont intéressées à la construction des modèles de tâches et ont mesuré les erreurs syntaxiques et sémantiques et le temps de réalisation du modèle. Ces études ont montré que la production d'un modèle de tâches semble être plus complexe que la compréhension d'un modèle déjà construit. Les concepteurs d'outils de modélisation doivent tenir compte de cette contrainte en réduisant la charge de travail mentale associée à la modélisation et ne pas surcharger visuellement les modèles. Dans les environnements ambiants, la modélisation des tâches présente certains besoins spécifiques [7]. Le modèle doit premièrement inclure la possibilité d'étiqueter une tâche selon des contraintes spatiales, deuxièmement permettre de spécifier les ressources physiques nécessaires à sa réalisation, troisièmement déterminer le niveau de granularité de la tâche à modéliser.

La principale caractéristique d'un tel modèle est qu'il doit être dynamique. En effet, ce modèle doit pouvoir être mis à jour au moment de l'exécution afin de suivre le déroulement des tâches au moment où elles sont accomplies.

Exploitation du modèle de tâches à l'exécution

Un grand nombre de modèles de tâches a été développé, en particulier dans le contexte des interfaces graphiques : HTA [2], GTA [33], CTT [20], UAN [30], TKS [10] [9], DIANE + [32], TOOD [15] et K-MAD [5]. Une étude comparative décrite dans [13] a montré que certaines propriétés doivent être prises en charge dans les modèles comme : la description des buts, une structure hiérarchique pour la modélisation des tâches, l'intégration d'opérateurs temporels décrivant le séquencement des tâches, et les détails des objets et des actions qui permettront la modélisation des interfaces utilisateurs. Cette étude a montré que plus les modèles de tâches sont expressifs plus ils deviennent complexes. Du moins complexe et le moins expressif au plus riche on trouve : HTA, MAD, GTA, TKS, CTT, Diane+, TOOD.

Dans le domaine de la génération automatique d'interfaces utilisateur, Klug et Kangasharju [12] présentent un modèle de tâches exécutable exploité pour générer une interface graphique dynamique composée de blocs d'interface préprogrammés. Ils étendent quelques notations de CTT pour permettre l'exécution dynamique du modèle de tâches en runtime en donnant quelques états dynamiques aux tâches feuilles.

Dans [36], les auteurs comparent les concepts de tâches dans deux langages de modélisation (développés comme des extensions d'un même modèle de tâches, CTT) qui sont UsiXML [14] et CTML [37]. UsiXML permet la création de différentes versions d'une interface adaptées à différents contextes. Il réutilise les opérateurs temporels standard de CTT et propose d'affiner l'opérateur ≪ Choice ≫ [29] [28]. On peut cependant noter que si CTT supporte l'utilisation informelle de pré-conditions des tâches, cela n'est pas exprimable dans UsiXML. CTML, quant à lui, est un langage de modélisation de tâches coopératives qui a été développé pour les environnements ambiants. Il permet de prendre en considération la dépendance de la localisation des objets et la modélisation des périphériques invoqués. L'avantage de CTML est qu'il utilise des pré-conditions et des effets évalués sémantiquement.

Nous avons constaté que CTML mis à part, aucun des modèles de tâches existants ne correspondait parfaitement aux exigences de la modélisation des tâches dans les environnements ambiants. Mais ce dernier n'exploite pas les performances temporelles des tâches et ne permet pas d'assister l'utilisateur. De plus l'éditeur et le simulateur de CTML ne sont pas disponibles publiquement, ce qui empêche sa réutilisation.

Plutôt que de proposer un nouveau modèle de tâches ex nihilo, nous avons préféré partir d'un modèle existant et l'étendre. Notre choix s'est porté sur le modèle CTT. Il intègre un ensemble riche d'opérateurs temporels ; il offre la possibilité de définir des pré-conditions et des post-conditions d'une tâche selon une certaine syntaxe (que nous nous proposons de raffiner par la suite) et il propose un éditeur accessible qui permet de saisir le modèle de tâches et de vérifier sa cohérence (CTT Environment ou CTTE) [16].

UN MODÈLE DE TÂCHES EXPLOITABLE EN RUNTIME

Dans cette section nous présentons les différentes extensions apportées à CTT dans notre modèle.

Transformation 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 qui se présente uniquement sous la forme d'un arbre binaire sans liens horizontaux entre les tâches : les liens horizontaux de CTT, qui représentent les opérateurs temporels, donnent lieu à de nouvelles tâches abstraites intermédiaires correspondant à ces opérateurs. Cette transformation est introduite dans le but de simplifier l'algorithme de suivi du déroulement qui n'aura plus qu'à s'appliquer sur un arbre binaire classique. 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 élémentaires qui peuvent être détectées par les capteurs de l'environnement, lors de leurs exécutions. Les tâches abstraites reflètent le comportement d'un opérateur temporel reliant deux sous-tâches, comme illustré sur la figure 1 :

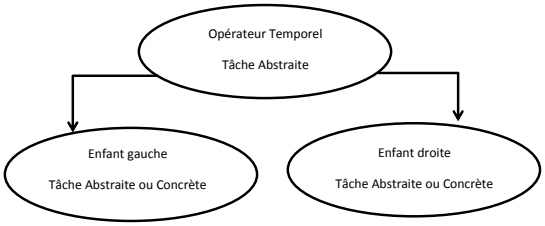

Figure 1. Exemple de décomposition de tâches.

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

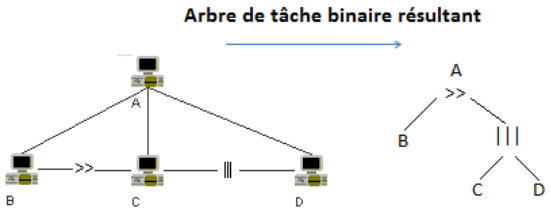

Figure 2. Arbre de tâches binaire résultant de la transformation.

Annotation des tâches élémentaires

Une tâche élémentaire 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).

Les propriétés statiques d'une tâche sont de deux types.
- On retrouve certaines propriétés qui ont été déjà introduites dans CTT et qui sont :
  - Son identifiant ou son nom.
  - Son importance (élevée, moyenne, basse).
  - Son caractère obligatoire : détermine si la réalisation de la tâche est indispensable pour atteindre un objectif donné.
  - Sa durée maximum de réalisation.
  - Sa durée minimum de réalisation.
- On aura aussi un certain nombre de propriétés que nous avons raffinées ou introduites pour les besoins propres de notre modèle et qui sont :
  - Service : représente un service invoqué par la tâche. On introduit ici la notion de service abstrait pour désigner un processus qui représente une suite finie de services destinée à atteindre un but donné.
  - Pré-condition : Une condition qui doit exister ou doit être établie avant que la tâche ne puisse se produire.
  - Post-condition : Constitue un ensemble de conditions dont on sait qu'elles doivent être vérifiées après l'exécution de la tâche. Les pré/post conditions peuvent contenir une information sur principalement trois grandes catégories de composantes à savoir :
    - l'environnement, par exemple la température
    - l'état des dispositifs du système
    - l'utilisateur : ses profils statique et dynamique. Par exemple le sexe sera une caractéristique statique alors que la position de l'utilisateur sera une caractéristique dynamique puisqu'elle peut changer au cours de l'utilisation du système.

  Les pré-conditions utilisées dans notre modèle sont écrites selon une syntaxe que nous avons définie et sont évaluées au moment de l'exécution. Une pré-condition est une formule en logique propositionnelle (dont les différentes parties sont reliées par les opérateurs logiques « OU », « ET » et « NON »).

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

Les opérateurs temporels

Les opérateurs temporels utilisés dans notre modèle sont ceux de CTT. Nous listons dans ce qui suit les différents opérateurs, selon leur ordre de priorité, en commençant par le plus prioritaire : Choice [], Order Independent |=|, Interleaving |||, Synchronization |[]|, Disabling [>, Suspend-Resume |>, Sequential Enabling ≫.

Les états des tâches

Comme nous proposons un modèle de tâches dynamique, l'état des tâches 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 : cela signifie que temporellement, la tâche peut être effectuée dès que ses pré-conditions seront satisfaites.
- RÉALISABLE : la tâche peut être effectuée (ses pré-conditions sont satisfaites).
- ACTIVE : la tâche est en cours d'exécution.
- SUSPENDUE : cette tâche était en cours d'exécution (Active) lorsqu'une autre tâche a démarré et l'a suspendue. Sa réalisation sera reprise une fois que la nouvelle tâche sera finie. Le même cas s'applique pour une tâche pouvant être réalisée (Possible ou Réalisable) qui 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.

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

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

Un scénario dans les environnements ambiants

Afin de faciliter la compréhension de l'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. Il peut soit commencer par la préparation du déjeuner soit par le nettoyage de la maison. Chacune de ces deux tâches peut être interrompue par un appel téléphonique ou par l'arrivée de quelqu'un qui frappe à la porte.
Pour préparer le déjeuner, il doit commencer par la cuisson des pâtes : il doit tout d'abord prendre la casserole dans le placard, la remplir d'eau et la mettre sur le feu. Dix minutes plus tard, une fois que l'eau est bouillante, il devra mettre les pâtes dans la casserole, les laisser cuire pendant 15 minutes puis les égoutter.
Pendant que les pâtes sont en train de cuire, il devra préparer la sauce en commençant par éplucher un oignon. À ce moment, nous supposerons que le téléphone fixe qui est dans le salon commence à sonner. Jean devra alors suspendre sa tâche en cours pour répondre à l'appel. Une fois la communication terminée, il devra revenir à la cuisine, couper l'oignon en petits morceaux, prendre la poêle du placard et y mettre l'oignon. Il devra ensuite ajouter l'huile et mettre le mélange dans la cocotte puis cinq minutes plus tard y verser le contenu d'une sauce préalablement achetée en grande surface et laisser cuire pendant dix minutes. Nous supposerons que dix minutes plus tard le système détecte que la plaque de cuisson n'a pas été éteinte. Le système devra alors indiquer à Jean d'enlever la sauce de la plaque de cuisson et de l'éteindre. Jean devra ensuite mélanger les pâtes avec la sauce. Une

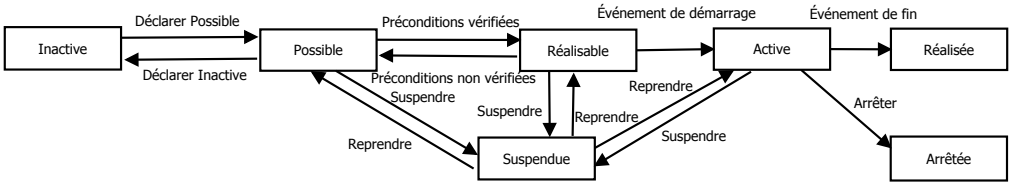

Figure 3. Les états possibles d'une tâche sous forme d'une machine à état finis.

fois le déjeuner prêt, il devra s'occuper du nettoyage de la maison.

Architecture globale de la solution

Le principe de notre approche est de comparer, au moment de l'exécution des tâches, les informations contenues dans le modèle de tâches à l'activité réelle de l'utilisateur afin de pouvoir lui apporter une assistance en cas de besoin. La figure 4 illustre cette approche.

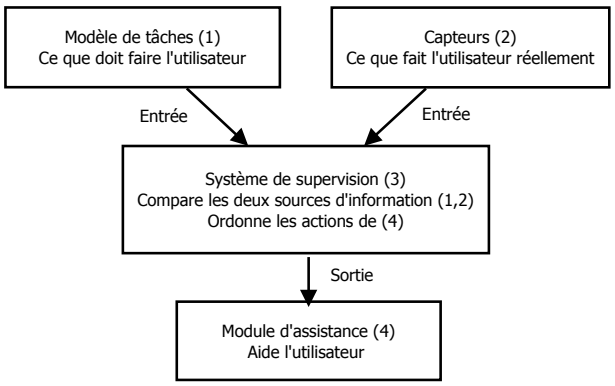

Figure 4. Approche globale.

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 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. En ce qui concerne le démarrage des tâches, nous ne traitons pas cette problématique pour le moment. Nous supposons que le démarrage est détecté en s'appuyant sur les travaux existants qui portent sur la reconnaissance de l'activité en cours, à titre d'exemple [8][31]. La détection de la fin des tâches est quant à elle effectuée grâce à l'utilisation des post-conditions affectées aux tâches. La vérification de ces post-conditions indique la fin de la tâche concernée. Par exemple la vérification de la fin de la tâche ≪ éteindre poêle ≫ revient à recevoir une information du capteur de température 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, il peut décider d'appeler le module d'assistance (4) afin d'informer l'utilisateur que telle sous-tâche n'a pas été faite correctement ou qu'elle a été omise. Dans le cas où l'utilisateur oublie d'éteindre la plaque, par exemple, le système détecte ce dysfonctionnement et appelle le module d'assistance qui indiquera à l'utilisateur de faire cette tâche.

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 (figure 5) :

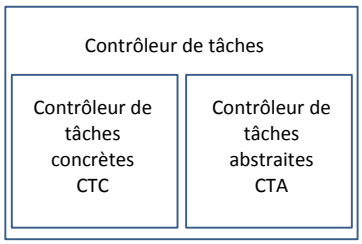

Figure 5. Schéma du contrôleur de tâche.

- Le contrôleur de tâches concrètes ou CTC : le CTC s'occupe du contrôle des tâches concrètes 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 selon des algorithmes différents dépendant du type de tâche abstraite, car chaque type de tâche abstraite correspond à un opérateur temporel différent.

Le CTC

Le CTC ≪ surveille ≫ les actions de l'utilisateur et invoque le module d'assistance dans un des deux cas suivants :

1. l'utilisateur fait une tâche qu'il ne devrait pas faire ;
2. l'utilisateur fait la bonne tâche mais pas comme il faut.

Notons qu'il existe un troisième cas où le module d'assistance peut être invoqué (≪ l'utilisateur ne fait pas une tâche qu'il devrait faire ≫) mais ce cas est pris en charge par le CTA. Dans ce qui suit nous allons expliciter comment le CTC gère chacun de ces deux cas. Considérons tout d'abord la figure 6 qui détaille le cycle de vie nominal d'une tâche (sans suspension ou arrêt). On définit ≪ le temps d'attente maximum d'activation ≫ comme la durée maximale prévue entre le moment où la tâche devient possible et le moment où

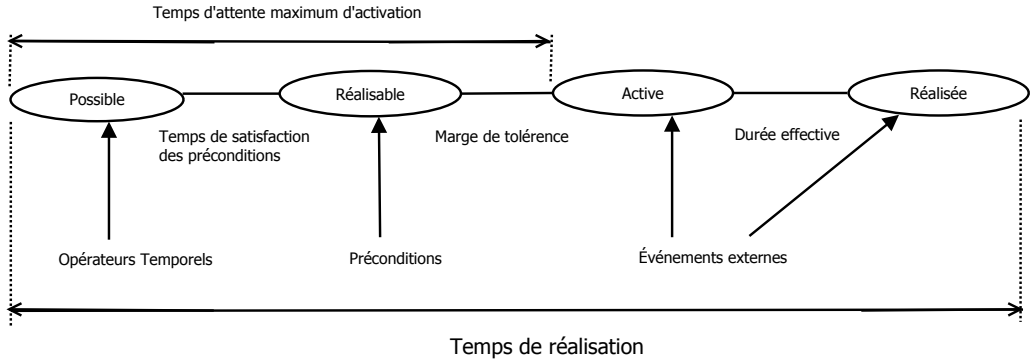

Figure 6. Cycle de vie nominal d'une tâche.

l'utilisateur commence concrètement sa réalisation. ≪ Le temps de réalisation ≫ correspond à la durée maximale prévisionnelle dédiée à la réalisation de la tâche. La figure 6 illustre ces différents états et les temps les séparant, décrivant ainsi l'ordonnancement normal des états d'une tâche qui va servir de référence pour le suivi de cette tâche.

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 moment de la faire et/ou que ses pré-conditions ne sont pas vérifiées (tâche non réalisable). Le CTC peut détecter ce cas directement à partir des états dynamiques des tâches. Par exemple, dans notre scénario ce cas peut arriver dans le cas où Jean essaye de mettre les pâtes dans l'eau alors que celle-ci n'est pas encore bouillante.

2. 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 les deux critères suivants :
   - L'effet : une tâche réalisée mais qui ne produit pas les effets attendus sera donc considérée comme une tâche n'ayant pas été effectuée correctement. Ce cas peut se présenter dans notre exemple dans le cas où l'utilisateur a effectué la tâche ≪ éteindre la plaque de cuisson ≫ alors que celle ci reste à température élevée après un long moment.
   - Les modalités de réalisation : 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.

Le CTA

Le CTA a en charge la supervision temporelle globale des tâches selon deux axes :
- l'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 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'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 dépassé et si la tâche est toujours à l'état ≪ Possible ≫ ou ≪ Réalisable ≫ (voir figure 6). Pour le contrôle de la réalisation, nous expliquerons dans un premier temps, en détail, 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'ordonnancement possibles entre tâches (différents types d'opérateurs temporels).

On définit le temps 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 7). 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. 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 par cumul des durées prévisionnelles de ses tâches concrètes comme suit : $D_{\mathrm{MAX}}(t) = \sum_{t' \in C} D_{\mathrm{MAX}}(t')$;

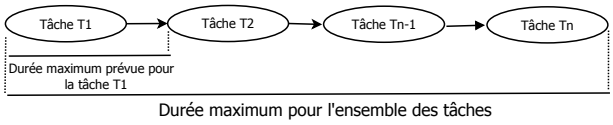

Figure 7. Temps maximum attribué à un ensemble de tâches.

Les 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

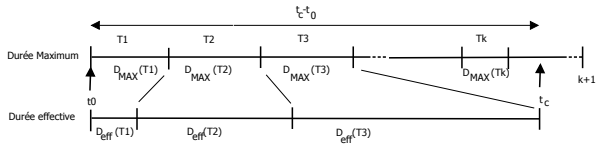

Figure 8. Temps maximum et effectifs d'un ensemble de tâches

Le CTA construit pour l'ensemble des tâches de la chaîne un tableau temporaire 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. Le résultat est illustré sur la table 1. Le CTA peut alors utiliser l'algorithme suivant :

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

Deux stratégies d'intervention peuvent alors être considérées :

1. Stratégie intrusive ou stratégie pessimiste : dès qu'il y a un dépassement des durées maximales prédéfinies le système doit produire des alertes (le cas 2).
2. Stratégie moins intrusive ou stratégie optimiste : ici on considère que bien que l'utilisateur ait pris du retard sur la réalisation des tâches de 1 à k la situation peut être rattrapée, donc on 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. On ne génère donc des alertes système à l'utilisateur que dans le cas 2b.

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 calculs des temps de début et de fin au plus tard sur lesquels va se baser notre système de supervision, nous allons propager les calculs sur notre structure en arbre. Nous commençons par voir de près les opérateurs temporels restants. Nous distinguons deux classes d'opérateurs :

1. Opérateur temporel à une seule possibilité de réalisation : l'opérateur ≪ Synchronization ≫. Il existe une seule manière possible pour réaliser les deux tâches : démarrer leur réalisation en même temps et la finir en même temps. La durée maximale (ou minimale) pour une tâche mère de ce type est approximée à la durée maximale (ou minimale) prévue pour ces deux tâches filles.
2. Opérateurs temporels avec un choix qui se fait en runtime, qui eux même peuvent être partagés en deux groupes :
   (a) ceux qui ont leur deux tâches filles obligatoires : ≪ Interleaving ≫ et ≪ Order Independent ≫. La durée maximale (ou minimale) pour une tâche mère de ce type est approximée à la somme des durées maximales (ou minimales) prévues pour ces deux tâches filles.
   (b) ceux qui n'ont qu'une tâche obligatoire et l'autre tâche optionnelle (pas forcément réalisée) qui à leur tour peuvent être partagé en deux groupes :
      i. nous ne connaissons pas par avance quelle tâche sera réalisée : cas de l'opérateur ≪ Choice ≫. La durée maximale d'une tâche mère de ce type est approximée à la durée maximale prévue pour sa tâche fille qui prend le plus de temps (de même pour l'approximation de la durée minimale mais cette fois

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

Table 1. Calcul des instants de début et de fin au plus tard

en considérant les durées minimales prévues pour les deux tâches filles).

ii. nous connaissons par avance quelle tâche sera réalisée (partiellement ou entièrement mais nous sommes sûrs de commencer avec cette tâche) : « Suspend Resume » et « Disabling ». Ici pour approximer la durée maximale ou minimale de la tâche mère deux stratégies peuvent être considérées :

A. 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 (en tenant compte aussi bien des durées des tâches qui ont forcément lieu que des tâches optionnelles).

B. 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. Nous mettons à jour les calculs en tenant compte des durées des tâches optionnelles si elles ont lieu concrètement.

SIMULATION D'UN SCÉNARIO

Le simulateur

Pour valider notre approche, nous avons développé un simulateur pour l'assistance à l'utilisateur et le suivi de ses tâches dans les environnements ambiants. Ce simulateur prend en entrée le fichier issu de la conception d'un modèle de tâche à l'aide l'éditeur CTTE [16]. Cet outil peut donc être utilisé par le concepteur pour saisir son modèle de tâche comme illustré sur la figure 9. Le modèle résultant est intégré directement dans notre simulateur.

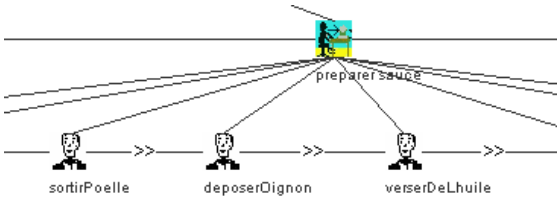

Figure 9. Exemple d'une partie d'un arbre saisi avec CTTE

Pour chaque tâche, un ensemble d'informations est saisi (son identifiant, sa fréquence, ses durées maximum et minimum de réalisation,...). En ce qui concerne l'ensemble de propriétés qui ne sont pas prises en compte dans le modèle CTT (les services appelés, le temps maximum d'attente d'activation et l'ensemble de ses pré et post-conditions), le concepteur peut utiliser le champ de texte « description » dans CTTE. Le simulateur analyse le 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.

La figure 10 montre un exemple du modèle de tâche 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. 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.
  - la barre de progression se remplit en fonction du temps écoulé lors de la réalisation de la tâche. La couleur de remplissage est bleue si on n'a pas encore dépassé la durée de réalisation qui a été prévue 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.

Déroulement du scénario

Le simulateur que nous avons développé permet au concepteur de simuler la production d'informations provenant des capteurs de l'environnement. Pour la tâche « SortirPoêle » les pré-conditions qui lui ont été assignées sont « placardOuvert » et « PoêleDansLePlacard » et

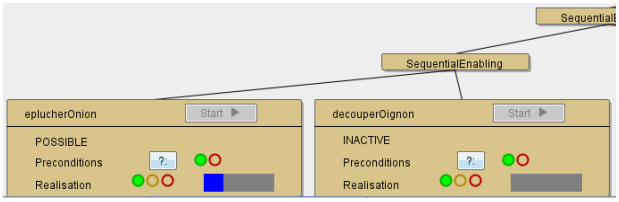

Figure 10. Notre arbre de tâches binaire

sont reliées par un « ET » logique. 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 en runtime et elle devient réalisable comme sur la figure 11. Il devient alors possible de cliquer sur bouton « Start » relatif à la tâche pour simuler la détection de son démarrage.

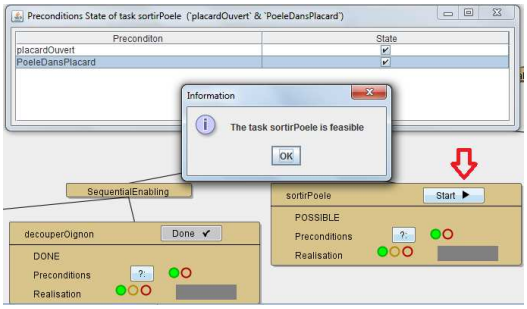

Figure 11. Vérification des pré-conditions d'une tâche qui devient faisable

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 12.

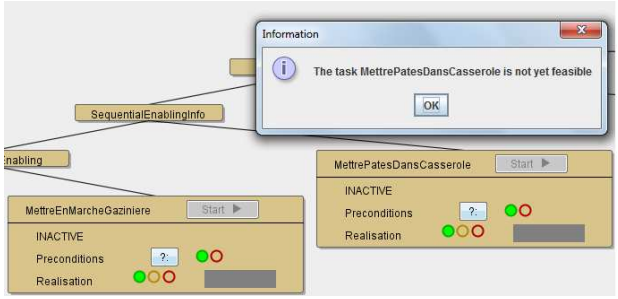

Figure 12. le CTC notifie une tâche non réalisable voulant démarrer

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 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 13.

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 14. De plus il ne restera plus de temps pour finir le

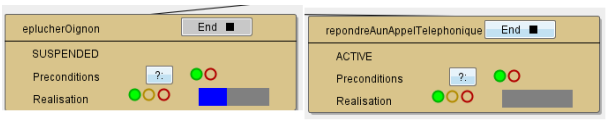

Figure 13. L'activité en cours suspendue par un appel

reste des tâches même en les effectuant aux performances maximales (avec les temps minimums prédéfinis) et donc le voyant rouge de réalisation s'allume comme indiqué sur la figure 14.

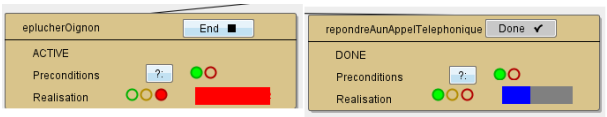

Figure 14. Reprise de la tâche suspendue suite à la fin de l'appel et notification par le CTA du retard engendré

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 de gérer dynamiquement l'état de chaque tâche et 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 pré-conditions...).

Nous avons également montré comment un système de suivi et d'assistance peut exploiter un tel modèle de tâches dynamique 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 via les informations fournies par les capteurs. Nous avons détaillé ensuite le fonctionnement du système de supervision qui intervient afin d'alerter l'utilisateur.

Enfin nous avons présenté une validation 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, 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. Il sera intéressant d'étudier l'application de l'approche à des domaines autres que l'informatique ambiante.

BIBLIOGRAPHIE

1. Annett J. Hierarchical task analysis. *Handbook of Task Analysis fir Human Computer Interaction* (2004), 67–83.
2. Annett J. & Duncan K. Task analysis and training design. *Occupational Psychology 41* (1967), 211–227.

3. Annett J., Duncan K. D., Stammers R. B. & Gray M. J. Task analysis. *London: Her majesty's stationery office.* (1971).
4. Balbo S., Ozkan N. & Paris C. Choosing the right task-modeling notation a taxonomy. *In D. Diaper and S. Neville (Eds.), The Handbook of Task Analysis for Human Computer Interaction* (2004), 445–465.
5. Baron M., Lucquiaud V., Autard D. & Scapin D. K-made : un environement pour le noyau du modele de description de l'activite. *18eme Conference Francophone sur l'Interaction Homme-Machine (IHM'2006)* (2006), 287–288.
6. Ducatel K., Bogdanowicz M., Scapolo F., Leijten J. & Burgelman J.-C. Scenarios for ambient intelligence in 2010. *Final report, Information Society Technologies Advisory Group (ISTAG)* (2001).
7. Gharsellaoui A., Bellik Y. & Jacquet C. Requirements of task modeling in ambient intelligent environments. *International Conference on Ambient Computing, Applications, Services and Technologies* (2012), 71–78.
8. Giersich M., Forbrig P., Fuchs G., Kirste T., Reichart D. & Schumann H. Towards an integrated approach for task modeling and human behavior recognition. *HCI 2007 4550* (2007), 1109–1118.
9. Johnson P. Human-computer interaction: Psychology, task analysis and software engineering. *London: McGraw-Hill* (1992).
10. Johnson P., Diaper D. & Long J. Tasks, skill and knowledge: Task analysis for knowledge based descriptions. *In Proceedings of First IFIP Conference on Human-Computer Interaction (Interact '84) 1* (1984), 23–28.
11. Kindberg T. & Fox A. System software for ubiquitous computing. *IEEE Pervasive Computing 1* (2002), 70–81.
12. Klug T. & Kangasharju J. Executable task models. *TAMODIA 2005* (2005), 119–122.
13. Limbourg Q. & Vanderdonckt J. Comparing task models for user interface design. *The Handbook of Task Analysis for Human-Computer Interaction* (2003), 135–154.
14. Limbourg Q., Vanderdonckt J., Michotte B., Bouillon L. & Lopez-Jaquero V. Usixml: A language supporting multi-path development of user interfaces. *Engineering Human Computer Interaction and Interactive Systems 3425* (2005), 200–220.
15. Mahfoudhi A., Abed M. & Tabary D. From the formal specifications of user tasks to the automatic generation of the hci specifications. *People and Computers XV* (2001), 331–347.
16. Mori G., Paterno F. & Santoro C. Ctte: Support for developing and analyzing task models for interactive system design. *IEEE transactions on software engineering 28* (2002), 797–813.
17. Ormerod T. C., Richardson J. & Shepherd A. Enhancing the usability of a task analysis method: a notation and environment for requirements specification. *Ergonomics,41(11)* (1998), 1642–1663.
18. Ormerod T. C. & Shepherd A. Using task analysis for information requirements specification: The sub-goal template (sgt) method. *The Handbook of Task Analysis for Human-Computer Interaction* (2004), 347–365.
19. Ozkan N., Paris C. & Balbo S. Understanding a task model: An experiment. *Proceedings of the HCI on People and Computers XIII* (1998).
20. Paterno F. Model based design and evaluation of interactive applications. *Berlin: Springer-Verlag* (1999).
21. Paterno F. Task models in interactive software systems. *Handbook of Software Engineering and Knowledge Engineering 1* (2002), 1–19.
22. Patrick J. Training : research and practice. *London: Academic Press.* (1992).
23. Patrick J., Gregov A. & Halliday P. Analysing and training task analysis. *Instructional science* (2000), 51–57.
24. Riva G. The psychology of ambient intelligence: Activity, situation and presence. *Ambient Intelligence: The Evolution of Technology, Communication and Cognition Towards the Future of Human-Computer Interaction (Emerging Communication)* (2005), 17–33.
25. Riva G., Loreti P., Vatalaro M., Vatalaro F. & Davide F. Presence 2010: The emergence of ambient intelligence. *Being There: Concepts, effects and measurement of user presence in synthetic environments* (2003), 60–81.
26. Sadri F. Ambient intelligence: A survey. *ACM: Computer Survey 43*, 4 (2011), 36–66.
27. Schulungbaum E. Model-based user interface software tools current state of declarative models. *Visualization and Usability Center* (1996).
28. Sinnig D., Chalin P. & Khendek F. Consistency between task models and use cases. *Proceedings DSV-IS 2007* (2007).
29. Sinnig D., Wurdel M., Forbrig P., Chalin P. & Khendek F. Practical extensions for task models. *TaMoDia 2007* (2007), 42–55.
30. Siochi A. & Hartson H. Task-oriented representation of asynchronous user interfaces. *CHI '89 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1989), 183–188.
31. Tapia E. M., Intille S. S. & Larson K. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing* (2004), 158–175.
32. Tarby J.-C. & Barthet M.-F. The diane+ method. *Computer-aided design of user interfaces* (1996), 95–120.
33. VanderVeer G. C., VanderLenting B. F. & Bergevoet B. A. J. Gta: Groupware task analysis - modeling complexity. *Acta Psychologica 91* (1996), 297–322.
34. Weiser M. The computer for the twenty-first century. *Scientific American* (1991), 94–100.
35. Weiser M. Some computer science issues in ubiquitous computing. *Communications of the ACM 36*, 7 (1993), 75–84.
36. Wurdel M. & Forbrig P. A comparative study of task concepts in usixml and ctml and their applications. *First International workshop on USer Interface eXtensible Markup Language (UsiXML'2010)* (2010), 173–182.
37. Wurdel M., Sinnig D. & Forbrig P. Towards a formal task-based specification framework for collaborative environments. *Computer-Aided Design of User Interfaces VI* (2009), 221–232.