HAL
open science

Retour d'expérience sur KISS, un outil de développement d'habitat intelligent par l'utilisateur final

Emeric Fontaine, Alexandre Demeure, Joëlle Coutaz, Nadine Mandran

To cite this version:

Emeric Fontaine, Alexandre Demeure, Joëlle Coutaz, Nadine Mandran. Retour d'expérience sur KISS, un outil de développement d'habitat intelligent par l'utilisateur final. Ergo'IHM 2012 - Ergonomie, Interaction Homme-Machine et Ingénierie Système, Oct 2012, Biarritz, France. hal-00779660

HAL Id: hal-00779660

https://inria.hal.science/hal-00779660

Submitted on 22 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Retour d'expérience sur KISS, un outil de développement d'habitat intelligent par l'utilisateur final

Emeric Fontaine, Alexandre Demeure, Joëlle Coutaz, Nadine Mandran
Université de Grenoble, Laboratoire LIG
BP 53, 38041 Grenoble Cedex
{emeric.fontaine, alexandre.demeure, joelle.coutaz, nadine.mandran}@imag.fr

RESUME
Nous présentons KISS (Knit your Ideas Into Smart Spaces), un outil de développement d'habitat intelligent par l'utilisateur final. KISS permet de programmer son habitat en termes de phrases exprimées en langage pseudonaturel. La mise au point de ces programmes peut se faire *in situ* dans l'appartement réel ou dans sa réplique virtuelle. L'évaluation de KISS montre que les utilisateurs parviennent à programmer un scénario réaliste.

Mots clés
Programmation/développement par l'Utilisateur Final, Habitat intelligent, intelligence ambiante, KISS.

ABSTRACT
We present KISS (Knit your Ideas Into Smart Spaces), an end-user development system for the home. KISS enables users to program their home with sentences expressed in a pseudo-natural language. Programs can be tested either with the virtual home or in the real home. In this paper, we describe KISS and report an evaluation that shows that users are able to program a real-life scenario.

Author Keywords
End-user Programming/Development (EUP), Smart habitat, ambient intelligence, KISS.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Human Factors; Design.

INTRODUCTION

Nous convenons de définir un habitat intelligent (*smart habitat*) comme suit : un lieu d'activités constitué de dispositifs et de fonctions interconnectés, capable de garantir en toute circonstance les services attendus par l'habitant avec une exigence de valeur comme le confort, l'économie d'énergie et plus généralement, le bien-être.

Deux écoles de pensée s'affrontent dans la mise en application de cette définition : soit l'habitat est le compagnon qui devance les besoins et désirs de l'utilisateur, soit l'habitat est un outil avec lequel l'utilisateur interagit de manière explicite [3, 26]. Dans les deux cas, il y a intelligence, donc adaptation avec respect de la valeur. Mais pour l'un, l'habitat agit de manière autonome avec le risque d'interaction incidente [9] ; pour l'autre, l'autonomie du système est placée sous le contrôle humain. D'un côté, l'utilisateur est un pur consommateur limité par les choix des concepteurs aussi centrés utilisateurs soient-ils ! De l'autre, l'utilisateur est le façonneur de l'intelligence de son habitat. Nous optons pour la seconde approche motivée par deux considérations : l'échec persistant de la domotique et le franc succès du « faîtes-le vous-même » (*Do It Yourself* ou *DIY*).

Les freins au développement de la domotique incluent le coût financier, l'absence d'interopérabilité entre les multiples technologies propriétaires mais aussi les difficultés d'installation et d'usage. En particulier, l'installation et son paramétrage requièrent des professionnels qu'il faut faire intervenir à chaque panne et modification du système. Dans ces conditions, redonnons le pouvoir à l'utilisateur. Deux voies complémentaires sont envisageables : les boutiques en ligne (*app stores*) et le Développement par l'Utilisateur Final (DUF).

Avec la voie des boutiques en ligne, l'habitat est vu comme un serveur central doté de dispositifs périphériques hétérogènes. L'offre consiste à fournir les bonnes abstractions, à masquer notamment la diversité des dispositifs techniques [10]. La production de petites applications ciblées, disponibles en ligne, s'en trouve favorisée. Dès lors, l'utilisateur a quelque contrôle sur son habitat puisque lui revient le libre choix des « applications à installer soi-même ». L'utilisateur n'en reste pas moins réduit au rôle de consommateur, dépendant d'une communauté de développeurs qui n'est pas à son service.

L'autre voie, celle du DUF, doit permettre la pleine créativité. KISS est un prototype DUF conçu dans cet esprit. Dans cet article, nous précisons les concepts clefs et analysons l'état de l'art en matière de DUF pour l'habitat intelligent. Nous présentons ensuite KISS et son apport suivis des résultats de l'évaluation expérimentale réalisée avec 13 sujets en condition semi-contrôlée dans l'appartement intelligent Domus du laboratoire LIG.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Ergo'IHM 2012, October 16–19, 2012, Biarritz, France.
Copyright 2012 ACM 978-1-4503-1015-4/12/05...$10.00.

PROGRAMMATION ET DEVELOPPEMENT PAR L'UTILISATEUR FINAL

Définitions

La programmation par l'utilisateur final (PUF) correspond à la situation où les rôles de programmeur et de consommateur de programme sont assurés par la même personne. L'objectif premier de l'utilisateur final n'est pas d'apprendre à programmer mais de résoudre de manière automatisée un problème personnel qui serait plus coûteux à résoudre sans l'aide de la machine [23].

Alors que le terme PUF concerne la phase de création de programme, le DUF désigne l'ensemble des méthodes, techniques et outils qui permettent à l'utilisateur final agissant en tant que développeur, de créer, de modifier ou d'étendre un logiciel [20]. Le Génie Logiciel par l'Utilisateur Final (GLUF) va plus loin avec la prise en compte du processus qualité [19]. De manière orthogonale, émerge la production sociale de solutions logicielles.

Ces définitions étant posées, notre analyse de l'état de l'art cible les outils DUF portant sur l'habitat intelligent.

État de l'art des outils DUF pour habitat intelligent

Notre analyse de l'état de l'art s'appuie sur trois critères complémentaires : (1) le langage de programmation et l'éditeur de programme dont la nature et la qualité ont un impact sur l'investissement attentionnel [5] ; (2) l'aide à la mise au point, nécessaire à toute activité de programmation ; (3) nature du couplage entre phase de développement et phase d'exécution, un sujet fort en pointe en génie logiciel au regard de l'adaptation dynamique, mais aussi au regard du comportement observé de l'utilisateur final chez lequel l'expression des requis, les activités de conception, de spécification, de codage et de mise au point, sont intimement imbriquées [19].

Nous avons retenu deux types de propositions : (a) les outils DUF émanant de la communauté IHM conçus selon une approche centrée utilisateur : a CAPpella [7], iCAP [8], CAMP [29], AutoHAN [4] et, dans une certaine mesure, BYOB [1], une extension de Scratch [21] ; (b) les outils émanant des communautés agents [14] ou systèmes répartis et intergiciels : TeC [27] et Pantagruel [11] et son simulateur DiaSuite [17]. Si les premiers mettent l'accent sur l'utilisabilité des éditeurs et du langage de programmation, les infrastructures d'exécution étant ad-hoc, les seconds trouvent leur point fort dans une infrastructure d'exécution à agents ou à composants orientés services, les aspects IHM étant alors placés au second plan.

Concernant les langages de programmation et les outils d'édition, la plupart des solutions s'appuie sur les langages visuels, rarement sur la langue pseudonaturelle (CAMP), tous manipulables via des dispositifs d'interaction usuels (écran, souris, clavier). Les Media Cubes d'autoHAN font figure d'exception dans ce paysage WIMP puisqu'ils sont les seuls à relever des IHM tangibles : comme pour les Siftables [22], l'utilisateur programme en plaçant côte à côte des cubes physiques, les Media Cubes. Toutes les propositions font usage de traits stylistiques qui facilitent la lecture des programmes, de même la compréhension de leur structure, mais seul iCAP en permet la personnalisation. Bien que la plupart des éditeurs s'appuie sur un langage visuel, la 3D n'a pas encore été considérée.

Les éditeurs sont massivement monosyntaxes à l'exception d'AutoHAN qui n'assure toutefois qu'une égale opportunité partielle : s'il est possible de produire un programme par composition de Media Cubes et d'en obtenir une représentation numérique sémantiquement équivalente dans une autre syntaxe (notamment, la syntaxe visuelle Vseq), il est impossible de faire le chemin inverse. Les Media Cubes, qui sont des dispositifs physiques, ne sont pas (encore) des mini-robots autonomes qui se déplaceraient pour fournir la version tangible du programme source exprimé en Vseq.

Concernant les paradigmes de programmation, nous observons peu d'exemples d'approche impérative (BYOB). La tendance s'oriente clairement vers des langages déclaratifs à base de règles ou d'événements. Il s'agit majoritairement de spécification donnant lieu à une compilation classique, exception faite pour a Cappella qui génère du code par généralisation. Le code généré est généralement inaccessible à l'utilisateur final. Seul AutoHAN donne accès en lecture comme en écriture à la notation XML, Lingua Franca, qui lui sert de langage pivot. Dans [14], le langage pivot est un langage de règles ECA. Notons aussi pour CAMP l'accès en lecture au code généré (en l'occurrence, une phrase en langue pseudonaturelle).

Dans tous les cas, nous constatons une séparation nette entre les phases de conception et d'exécution à l'exception de TEC qui s'appuie sur une infrastructure d'exécution à composants orientés services dynamiques. Peu d'efforts ont porté sur l'aide à la mise au point. Les idées telles que la Whyline [18] ou le "What You See Is What You Test" (WYSIWYT) [12] appliqué aux tableurs, n'ont pas été considérés. La possibilité de tester son programme dans le monde réel ou dans un monde dual simulé est rarement possible. Par exemple, a Cappella, non seulement ne rend pas observables les règles qu'il a apprises, mais n'autorise les tests que dans le monde réel. Dans ces conditions, comment tester l'effet d'un événement susceptible de se produire dans plusieurs mois ?

KISS traite un sous-ensemble de ces limitations. Il vise à redonner le pouvoir à l'utilisateur non seulement à l'exécution du système mais aussi en amont, dès la phase de conception. Ses points-clefs : langage pseudonaturel déclaratif par règles spécifiées au moyen de menus contextuels, mise au point dans les mondes réels et virtuels, couplage étroit entre phases de conception et d'exécution.

KISS (KNIT YOUR IDEAS INTO SMART SPACES)

KISS comprend un éditeur de phrases, un simulateur d'habitat (réplique numérique de l'habitat réel) pour la mise

au point et une horloge pour la manipulation du temps.

On trouvera une description détaillée de la réalisation technique de KISS dans [13]. En synthèse, les phrases écrites par l'utilisateur final au moyen de l'éditeur sont traduites en scripts interprétés par une infrastructure d'exécution à composants orientés services. Ces scripts expriment l'adaptation de l'habitat intelligent en termes de *{situation-action}* où *action* spécifie le plan de reconfiguration dynamique des composants logiciels que l'infrastructure d'exécution doit réaliser lorsque se présente la *situation* en question. Dans cette infrastructure, tout appareil (lampe, capteur, etc.) est modélisé sous forme de deux dispositifs UPnP, l'un correspondant à l'appareil réel, l'autre à son dual virtuel. Ainsi, tout dispositif UPnP est métadécrit par son type, sa localisation et son monde d'appartenance (réel ou virtuel). L'infrastructure, par le biais du protocole UPnP, scrute les appareils (réels et virtuels), les mémorise dans sa base de connaissances, agit sur les effecteurs (réels et virtuels) de l'habitat (réel ou virtuel) en conformité avec la sémantique des phrases sources traduites en scripts *{situation, action}*.

L'éditeur de phrases

Le choix du langage pseudonaturel (LPN) à base de règles, dérivé du français, est motivé par l'évidente compétence en langue française des utilisateurs cibles. Si un langage pseudonaturel a l'avantage de la familiarité, il n'en reste pas moins contraint. Dès lors, il est nécessaire de rendre observable à l'utilisateur débutant l'espace des possibles, selon les principes du *feedforward*[1]. Comme le montre la figure 1, nous avons retenu les menus contextuels mais appliqués ici au langage naturel comme dans [28]. La zone en haut à gauche porte sur la spécification de la partie Condition de la règle. En haut à droite, les menus pour la spécification de l'action. Dans les deux zones, l'utilisateur sélectionne, via des menus déroulants, les éléments terminaux des phrases. La sélection d'un élément de menu entraîne l'affichage dynamique d'un nouveau menu si la phrase peut être complétée (cf. http://iihm.imag.fr/demo).

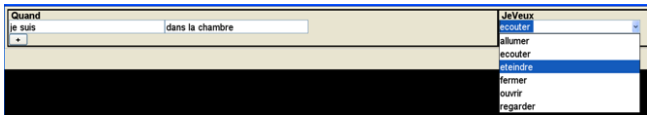

Figure 1 : Rédaction de phrase LPN au moyen de menus.

Le langage KISS est défini par une grammaire à plusieurs étages de façon à distinguer les généralités des spécificités, mais aussi les aspects statiques des aspects dynamiques de l'habitat (voir détail dans [13, chap.4]). En synthèse, une grammaire noyau définit la structure syntaxique commune à toutes les phrases en deux parties (condition et action), ce qui donne en syntaxe concrète : *Quand … je veux …* Comme le montre la Figure 2, *condition* et *action* utilisent le seul opérateur *et* (car plusieurs études ont démontré la mauvaise compréhension du *ou*, qui peut être exclusif ou traduire une union).

Une *condition* est une expression logique *et* de *facts* définis par la grammaire du contexte (*locationFact*, *timeFact*, etc.), ce qui permet d'exprimer *dans la chambre et il est entre 7h et 9h*. Une *action* est une conjonction d'*actionName* définis par la grammaire des actions (*open*, *watch*, etc.).

La grammaire des actions décrit la structure grammaticale de chaque action possible dans l'espace intelligent, ceci indépendamment des instances d'entités sur lesquelles ces actions s'appliquent mais aussi indépendamment du contexte. Ainsi, l'action *watch* admet comme complément d'objet direct un *observable*, c'est-à-dire une entité qui a la capacité d'être observée et un *viewer* comme complément circonstanciel de lieu. À leur tour, *observable* et *viewer* sont définis par la grammaire des classes abstraites.

La grammaire des classes abstraites est alimentée dynamiquement au moyen de requêtes SPARQL auprès de la base de connaissance RDFS de l'infrastructure d'exécution. Ainsi, dans notre exemple, l'analyseur syntaxique de KISS obtient dynamiquement tous les *observables* terminaux (tel l'*état de la machine à laver*) et tous les *viewers* terminaux (telle la *Télévision*) présents dans l'habitat. Ce sont ces terminaux que proposent de manière structurée les menus de la figure 1 en parfaite conformité avec les capacités de l'habitat, qu'il soit réel ou virtuel.

<u>Grammaire noyau</u>
sentence=condition action
condition=fact [operator condition]
action=unitaryAction [operator action]
operator="et "
unitaryAction=actionName
fact=!contextualInfoGram! *redirection vers gram. du contexte*
actionName=!actionGram! *redirection vers gram. des actions*
<u>Extrait de la grammaire du contexte</u>
fact=locationFact / timeFact / …
timeFact="il est " time
<u>Extrait de la grammaire des actions</u>
actionName= open / close / watch /…
open="ouvrir " opennable
watch="regarder" observable "sur " viewer
viewer = !abstractThingsBDC! *redir. vers gram. des classes abstraites*
observable=!abstractThingsBDC!

Figure 2 : Extraits de la grammaire ABNF de KISS (sans les traits sémantiques pour la génération des scripts)**.**

Les phrases ainsi rédigées sont listées dans un historique qui permet à l'utilisateur de les consulter et de les modifier à tout moment. À chaque ajout de phrase, l'utilisateur peut la tester soit dans l'habitat réel, soit dans l'habitat virtuel.

L'habitat virtuel comme metteur au point

L'habitat virtuel est une réplique de l'habitat réel. Pour notre expérimentation, nous avons utilisé l'appartement Domus (http://multicom.imag.fr/spip.php?rubrique89) du LIG. Domus est équipé de capteurs (présence, température,

[1] La reconnaissance vocale conviendrait toutefois à un utilisateur expert.

porte-serviette) et d'effecteurs (volets, lumières, cafetière, thermostat et radio). La Figure 3 montre l'IHM de l'habitat virtuel sous forme d'un plan 2D.

À gauche, la cuisine comprend deux lampes actuellement éteintes (ronds noirs) et une cafetière à l'arrêt (icône de cafetière barrée d'une croix). La salle d'eau est équipée d'un capteur de température (il fait actuellement 12°), d'une lampe éteinte et d'un porte-serviette (rectangle rouge). La chambre, pièce centrale de l'appartement, est celle où se trouve actuellement l'avatar (représentant de l'utilisateur). Elle contient deux lampes (actuellement éteintes), une radio (actuellement à l'arrêt) et un lit équipé de capteurs de pression (rectangle entouré de blanc). Au bord du lit, un bouton que l'utilisateur peut actionner en cas d'exception. Le fond clair de la chambre signifie que le capteur de présence de la chambre a détecté la présence de l'avatar. La pièce la plus à droite correspond au bureau que nous avons utilisé dans notre expérimentation comme lieu de programmation de l'habitat.

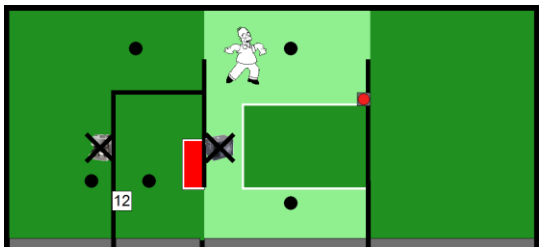

Figure 3 : l'IHM de l'habitat virtuel.

L'horloge

L'horloge permet de gérer le temps de référence. L'utilisateur peut : 1) visualiser la vitesse à laquelle s'écoule le temps ; 2) visualiser l'heure actuelle ; 3) régler le facteur d'accélération du temps, soit en le ramenant à 1, soit en le multipliant par 2 jusqu'à soixante fois plus rapide que dans la vraie vie ; 4) définir si l'heure de référence s'applique au monde virtuel, au monde réel ou bien aux deux à la fois.

La conception de KISS est le résultat de l'analyse de l'état de l'art présentée ci-dessus, mais aussi d'une étude de terrain qui démontre que l'utilisateur final est disposé à construire des assemblages de dispositifs et de services pour faciliter sa vie au quotidien [6]. Le scénario utilisé dans l'expérimentation est tiré de cette étude (voir figure 4).

TESTS POUR UNE APPROCHE DE CO-CONCEPTION

KISS, en tant que prototype, doit être soumis à l'appréciation de l'utilisateur final dans une approche de co-conception au sens de la conception centrée utilisateur. Les méthodes mises en œuvre sont des méthodes qualitatives qui permettent d'observer et de comprendre des phénomènes. Les données sont des enregistrements audio et vidéo ainsi que des réponses à un questionnaire. L'analyse des données est conduite selon les recommandations de Mucchielli [24] en analyse thématique.

L'objectif du test d'acceptabilité est double. D'une part, valider l'usage du langage pseudonaturel (LPN) comme moyen de programmer le comportement d'un habitat intelligent, d'autre part, valider l'utilité d'un appartement virtuel pour la mise au point des programmes. En particulier, il s'agit de vérifier que l'utilisateur peut exprimer aisément ce qu'il souhaite avec le vocabulaire fourni et qu'il comprend quels sont les objets sur lesquels porte le programme qu'il rédige et quelle en est la sémantique. L'utilisateur doit être capable de se repérer dans l'appartement virtuel et d'y retrouver les objets qu'il programme.

L'absence de capteurs de présence dans Domus nous a contraints à mettre en place un magicien d'Oz, le compère simulant ces capteurs en observant à distance les déplacements de l'utilisateur au moyen de caméras.

Le matin, vous avez l'habitude de vous réveiller à 7h. Vous aimez vous faire réveiller par une musique douce et qu'une lumière tamisée s'allume. Si vous restez au lit trop longtemps, vous aimeriez que l'intensité de la lumière et de la musique augmente. Quand vous êtes décidé à vous lever, vous appuyez sur le bouton et vous vous levez. La lumière change d'intensité. Le volume sonore passe à un niveau acceptable. Une fois que vous êtes levé, vous souhaitez que les volets s'ouvrent. Ensuite, vous allez dans la salle de bain, la lumière s'allume dès que vous rentrez dans la pièce. L'eau de la douche est préchauffée et une lumière projetée sur la douche vous indique la température de l'eau. Quand vous sortez de la douche, et commencez à vous sécher, la cafetière se déclenche de sorte que le petit-déjeuner soit prêt.

Figure 4 : Extrait du scénario.

Protocole

Le protocole expérimental comprend six étapes successives.

1) Le participant est accueilli par l'expérimentateur qui lui présente l'expérimentation en quelques mots.

2) L'expérimentateur lui fait visiter Domus. Il lui fournit un plan de l'appartement et lui montre, pièce par pièce, les différents équipements, y compris les capteurs de présence.

3) Le participant est formé à l'utilisation de KISS. Pour cela, l'expérimentateur rédige une phrase en LPN : *quand je suis dans la salle de bain, je veux allumer la lumière de la salle de bain avec une intensité moyenne*. Le participant est ensuite invité à se rendre dans la salle de bain pour constater l'éclairage. Quand il en sort, il s'aperçoit que la lumière reste allumée. L'expérimentateur en explique la raison et propose au participant de rédiger une phrase pour éteindre la lumière de la salle de bain quand il en sort. Le participant rédige la phrase correspondante et teste le résultat en entrant et sortant de la salle de bain.

L'expérimentateur propose ensuite d'ajouter une contrainte horaire à la phrase précédente. Il montre comment revenir sur une phrase déjà rédigée puis le guide pour reformuler la phrase avec des contraintes horaires : *quand je suis dans la salle de bain et quand il est entre 20h et 22h, je veux allumer la lumière de la salle de bain avec une intensité moyenne*. Pour tester cette phrase, le participant doit changer l'heure, ce qui permet à l'expérimentateur de lui

présenter l'horloge et de lui en expliquer le fonctionnement. Enfin, l'expérimentateur présente l'appartement virtuel. Il explique qu'il s'agit d'une réplique de Domus et y situe les pièces et équipements. L'expérimentateur propose de déplacer l'avatar dans la salle de bain, de le sortir, puis de refaire la même manipulation en changeant l'heure.

4) Le participant doit programmer à l'aide de KISS un comportement de l'habitat correspondant à un scénario du lever, moment clef d'une journée. Le scénario est inscrit sur papier et lu par l'expérimentateur. Nous avons pris soin de ne pas faire comporter trop de mots similaires entre la phrase lue et les mots proposés dans les menus déroulants de KISS. Le scénario comporte également des actions irréalisables dans cet appartement. De façon à observer le participant, l'expérimentateur reste présent durant toute la programmation, mais ne répond plus aux questions. Il l'encourage cependant à formuler à haute voix ses actions et interrogations de façon à suivre son raisonnement. Lorsque le participant a suffisamment confiance dans le programme qu'il a développé, il peut le tester dans l'appartement accompagné de l'expérimentateur.

5) L'expérimentateur vérifie avec le participant le bon déroulement du scénario. Si ce n'est pas le cas, il questionne le participant sur la raison de l'écart entre le comportement constaté de l'appartement et celui qui est attendu. Le participant apporte à son programme les modifications souhaitées et teste à nouveau.

6) Enfin, le participant est convié à un débriefing au cours duquel il fait part de ses impressions sur l'exercice qu'il vient d'effectuer et sur KISS. C'est également l'occasion pour l'expérimentateur de revenir sur les actions et interrogations du participant notées à l'étape 4. Enfin, avant de le libérer, le participant remplit un questionnaire d'évaluation portant sur KISS.

Ces six étapes durent au maximum 2 heures par participant et nécessitent la présence de deux expérimentateurs : l'un pour jouer le magicien, l'autre pour accompagner le participant durant le test.

Recrutement des participants

Pour cette expérimentation, nous avons recruté 13 participants par courriel et par relations personnelles directes et indirectes. L'échantillon comprend 6 femmes et 7 hommes, de 31 à 66 ans (moyenne d'âge 39 ans), avec source de revenu et répartis comme suit : 4 informaticiens (sujets S2, S3, S6, S8), 1 agent des impôts (S1), 1 ingénieur (S4), 1 avocate (S5), 1 pharmacien (S7), 1 conseillère à l'emploi (S9), 1 technicien régleur (S10), 1 maître de conférence (S11), 1 assistante sociale (S12), 1 chargée de communication (S13).

Résultats

Aspects positifs

Les participants ont globalement apprécié l'expérience. L'interaction avec l'habitat est « *amusante* » (S3, S6), « *sympathique* » (S9, S12), « *ludique* » (S13). Cette appréciation générale indique la volonté des utilisateurs finaux à contrôler leurs habitats intelligents.

Les participants pensent que l'utilisation de l'éditeur de phrases et de l'habitat virtuel est un bon moyen pour programmer l'appartement (10/13). Parmi les trois opposants (S1, S7, S13), S7 et S13 se contenteraient du LPN. Sept autres pensent que ce langage est suffisant mais que l'habitat virtuel est un bon complément pour programmer. Cela porte à 9 (sur 13) le nombre de participants d'accord pour affirmer que le LPN est suffisant pour programmer l'appartement. Seuls 4 participants ont besoin de l'habitat virtuel (S2, S5, S11, S12). Nous avons donc 3 catégories d'utilisateurs : a) Ceux qui ont besoin du langage pseudonaturel et de l'habitat virtuel ; b) Ceux qui n'ont besoin que du langage et c) Ceux qui ont besoin du langage mais pour lesquels l'habitat virtuel est un bon complément. Il est possible qu'un effet d'âge explique en partie l'existence de ces trois catégories. En effet, la moyenne d'âge de la catégorie a est de 47 ans alors que celles des catégories b et c sont respectivement de 34 et 35 ans.

Bien que plusieurs participants aient indiqué ne pas vouloir utiliser fréquemment l'habitat virtuel, son utilité a bien été comprise de tous : « *ça va me servir à vérifier que ça marche sans me déplacer* » (S1), « *j'aime bien rester assis sur ma chaise* » (S9), « *pas besoin de se lever* » (S8), « *on peut voir, revoir, et rejouer le scénario jusqu'à ce qu'il corresponde à ce que l'on veut programmer* » (S11). Enfin, tous les participants attestent que l'habitat virtuel permet d'identifier correctement les objets (13/13).

Le nombre de participants indiquant vouloir utiliser fréquemment le LPN est quant à lui plus élevé (9/13). Ce positionnement apparaît logique puisque seul ce langage a un réel impact sur l'appartement intelligent. Cependant, quatre participants (S2, S7, S8, S11) disent ne pas souhaiter utiliser le LPN fréquemment. Pour trois d'entre eux, la raison en est qu'ils veulent le faire "*une fois pour toute*" ou y revenir « *rarement* » (S8). Notons que deux autres participants ont exprimé oralement une remarque similaire bien que leurs réponses au questionnaire ne le montrent pas.

Le LPN a été perçu comme facile à utiliser (7/13). Seuls deux sujets l'ont trouvé inutilement complexe (S5 et S8). Cependant, pour S8, le résultat du questionnaire doit être mis en regard de ses commentaires : « *tout a été facile, ça se compliquerait s'il y avait d'autres options* » (S8). En d'autres termes, il semble que S8 anticipe la difficulté d'usage du LPN et de sa technique d'interaction par menus en cas de passage à l'échelle.

Tous les participants affirment que le vocabulaire utilisé permet de bien identifier les objets. *« À chaque fois, j'ai pu trouver ce que je voulais, même si ce n'était pas les mêmes mots que dans le scénario* » (S10). De même, ils sont 11 à

estimer que le paramétrage de l'intensité de la lumière était facile à réaliser.

Les participants se sentent capables d'utiliser le LPN sans l'assistance d'une personne technique (11/13). Ils pensent également que la plupart des utilisateurs peut apprendre à l'utiliser rapidement (11/13) : « *C'était accessible pour moi* » (S10), « *facile à comprendre* » (S6), « *compréhensible car c'est du langage parlé, c'est très vivant* » (S2). Ils indiquent ne pas avoir eu besoin de beaucoup de choses avant de pouvoir utiliser ce système (9/13) : « *Ça rapproche le raisonnement informatique à notre raisonnement à nous* » (S2), « *Ça exprime ce qu'on va faire naturellement, comme on se l'imagine* » (S9). *A contrario*, les principales difficultés rencontrées portent sur le LPN.

Difficultés rencontrées

Deux participants ont trouvé le LPN difficile à utiliser dans son ensemble (S5 et S11). Ils sont rejoints par sept autres participants quand il s'agit de poser des contraintes horaires. Seuls quatre participants (S1, S3, S4, S7), dont trois informaticiens, trouvent la pose des contraintes horaires aisée. Ce résultat est cohérent avec l'étude de C. Plaisant et al. [25].

Pour l'ensemble des participants (13/13), le LPN est cohérent : « *compréhensible, pas ambigu* » (S3, S4). Cependant, plusieurs sujets, dont S3, ont interprété de façon erronée, par rapport à l'implémentation proposée par notre système, le sens de certaines actions telles que : être réveillé et faire chauffer. Pour eux, être réveillé implique « *automatiquement* » qu'ils sont dans le lit (S3, S6, S8, S10) : « *car je dors dans mon lit* » (S10). Quant à l'action « *faire chauffer* », elle implique pour eux l'action « *regarder la température* » (S3, S4, S7).

Raisonnement non sequitur. Dans la même veine, nous avons observé que deux participants (S1 et S9) menaient des raisonnements non sequitur sur certaines conditions. Le participant (S9), qui est informaticien, explique que « *Quand je rentre*, il se passe une action, et quand je sors, l'action est toujours en cours. Alors que *quand je suis dans* implique : *tant que je suis dans* il se passe une action et *quand je n'y suis plus* l'action s'arrête ». Par exemple, « *Quand je suis dans la salle de bain, allumer la lumière de la salle de bain* » implique que « *Quand je ne suis pas dans la salle de bain, éteindre la lumière de la salle de bains* ». Ce type de raisonnement est aussi apparu lors de la spécification de règles portant sur l'horaire. En particulier, la phrase « *Quand il est entre 7h et 8h, allumer la lumière* » a été interprétée par S1 comme « *Tant qu'il est entre 7h et 8h, allumer la lumière, l'éteindre sinon* ».

Une explication possible à ces raisonnements réside peut-être dans la différence d'aspect des verbes. En effet, le verbe de la condition « *quand je rentre* » est d'aspect inchoatif alors les verbes des expressions « *je suis dans* » et « *il est entre 7h et 8h* » sont d'aspect progressif ou duratif. Les participants ont toujours bien interprété le cas inchoatif « *quand je rentre* » alors qu'ils ont parfois mal interprété (par rapport à l'intelligence de KISS et de son infrastructure) les cas progressifs/duratifs « *je suis dans* » et « *il est entre* ».

Vocabulaire limité. Trois participants ont trouvé le vocabulaire de LPN limité (S5, S11, S13). Par exemple, il n'est pas possible de dire les choses de façon globale comme « *éteindre toutes les lampes* » (S11), ou inversement de désigner un équipement parmi plusieurs identiques : « *les volets, les volets d'où ?* » (S5 rejoint par S13).

Le vocabulaire ne s'adapte pas à l'utilisateur. Une grande difficulté ressentie par les participants est de « *s'approprier un vocabulaire* » (S4) qu'ils ne connaissent pas ou dont la sémantique associée dans KISS est différente de la leur. Par exemple, l'action « *allumer* » est, dans KISS, l'unique moyen fourni à l'utilisateur pour modifier l'intensité des lumières. Lorsque les participants ont voulu programmer l'augmentation de l'intensité lumineuse, deux problèmes se sont posés à eux : trouver la bonne action (S6, S7, S10) car « *ce n'est pas l'action « allumer », car les lumières sont déjà allumées. Je veux juste augmenter* » (S7) ; et comprendre les différents niveaux d'intensité : « *ce n'est pas ma notion d'intensité* » (S13).

En raison de ces difficultés, certains participants ont eu le sentiment d'avoir fait des erreurs : « *L'appartement a compris, c'est moi qui ai du mal à dire ce que je voulais* » (S11), *« L'appartement a compris ce que je lui ai dit, c'est moi qui me suis trompé* » (S12), « *L'appartement a compris, j'ai fait des erreurs* » (S13), *« La maison a réagi à ce que j'ai dit, c'est ma faute, je me suis trompé* » (S6). Seul un sujet a exprimé le problème dans l'autre sens : « *C'est nous qui nous nous adaptons au système.* » (S9).

Les participants se sentent toutefois capables de surmonter ces difficultés avec le temps. *« Le premier coup est un peu galère, le second est génial* » (S5), « *après deux trois fois, ça va* »(S8), « *quand on est habitué, ça va* » (S12), « *À force de faire, j'y arriverai mieux* » (S13). Cet effet d'apprentissage a été ressenti par sept participants.

Ordonnancement des phrases mal compris. Certains participants ont cru que l'ordre des phrases avait une importance (S10). Par exemple, un sujet a souhaité déplacer une phrase entre deux autres, un autre explique pourquoi il n'a rédigé qu'une phrase avec un horaire : *« j'ai supposé que tout était lié, j'ai commencé avec un horaire, tout en découle* » (S2). D'autres participants auraient apprécié définir des contraintes d'ordre entre les phrases, « *j'avais envie de dire après* » (S11, S3), « *d'ajouter un délai de 15 minutes* » (S5, S7), « *de dire si je suis encore dans mon lit après que la règle 1 a été enclenchée* » (S7). Pour (S4) tout comme (S7), certaines phrases ont du sens, si et seulement si d'autres phrases ont été exécutées.

Complexité du paramétrage et de la gestion des règles. Plusieurs participants ont exprimé leur peur « *d'oublier*

quelque chose » (S3, S10, S11, S12). Pour l'un des participants, cela justifie la non-utilisation du LPN : « *je ne veux pas utiliser ça, car je pense qu'on oublierait toujours quelque chose* » (S11). Un autre participant pointe la nécessité d'être bien concentré pour ne rien oublier : *« il ne faut pas le faire à 11h du soir* » (S10). Cela paraît toutefois surmontable (S13) : « *prendre en compte tous les paramètres, c'est une gymnastique de l'esprit, mais ce n'est pas vraiment dur* ».

Bien que chaque participant n'ait pas rédigé plus de sept phrases, sept des treize participants ont rédigé deux fois la même phrase en pensant en rédiger deux différentes. En situation réelle, le problème risque d'être d'autant plus prégnant que le nombre de phrases à rédiger sera plus élevé.

Besoin d'un réel service rendu. Certains participants ont émis des réserves quant à l'intérêt d'un habitat intelligent ainsi que de sa programmation pour « *allumer trois lumières* » (S5). « *Si c'est pour allumer les lumières, j'appuie déjà sur un interrupteur, allumer les lumières automatiquement ne me fait pas gagner du temps. Par contre, pour quelque chose qui me fait gagner du temps sur des tâches que je ne veux pas faire (repasser, cuisiner), là ça a un intérêt* » (S1). Les utilisateurs sont donc prêts à passer du temps à programmer s'il y a valeur ajoutée.

Pistes d'améliorations

En premier lieu, il ressort de nos tests que le vocabulaire utilisé dans l'éditeur de phrases devrait être adaptable. Les participants auraient aimé ajouter leurs propres mots (S7). Ils ont éprouvé le besoin « *d'enrichir le vocabulaire* » (S4), d'ajouter leurs notions d'intensité (S3, S13), d'introduire leurs sémantiques (S1, S9). Certains ont souhaité avoir une zone de texte libre pour écrire ce que bon leur semblait (S1 et S2). Une piste envisageable est l'usage d'un vocabulaire co-construit [2] où l'utilisateur et le système apprennent l'un de l'autre comme dans TellMe [15] (au risque d'être disruptif par les questions posées périodiquement à l'utilisateur).

Explicitation de la sémantique. La mauvaise compréhension par l'utilisateur des phrases construites avec KISS est due principalement à une interprétation différente entre l'utilisateur et le système. Pour remédier à ce problème, nous pensons qu'il serait intéressant d'explorer l'utilisation de techniques de « *programmation avec exemple* » [16]. Dans cette optique, le système pourrait associer à chaque phrase une ou plusieurs représentations lui correspondant dans le simulateur. Par exemple, la phrase *« si je suis dans la salle de bain, allumer la lumière* » pourrait se traduire par une animation montrant l'avatar entrant dans une salle de bain, dont la lampe est éteinte, celle-ci s'allume, puis l'avatar sort et la lampe reste allumée.

Dans le même ordre d'idée, de nouvelles unités syntaxiques pourraient être introduites par l'utilisateur via une interface idoine pour désigner des groupes d'objets non prévus à l'origine. Rappelons-nous, des participants ont exprimé le besoin de « *tout éteindre quand ils sortent de la maison* » ou « *d'ouvrir les volets exposés sud* ».

Ordonner/grouper les phrases. Certains participants ont eu l'impression que l'ordre des phrases avait une importance : « *j'ai supposé que tout était lié, j'ai commencé avec un horaire, tout en découle* » (S2). Cette façon de voir les choses résulte probablement du fait que, dans le langage naturel, les phrases d'un texte s'assemblent pour exprimer un message. Dans notre approche, les phrases miment d'une façon trop proche l'expression d'une base de règles. Peut-être que l'utilisation de la notion de paragraphe permettrait de mieux articuler plusieurs phrases entre elles.

Plusieurs participants ont exprimé le besoin de regrouper des phrases, soit en fonction de leurs dépendances (S4, S7, S11) : « *grouper des étapes entre* » (S7), soit en fonction de conditions similaires (S2, S3), soit par fonctionnalités, par exemple les phrases liées à l'économie d'énergie (S5). Ces regroupements confirment le besoin de paquets de services déjà observés lors de notre étude de terrain [6]. De plus, plusieurs participants ont émis l'idée que des groupes de phrases pourraient être plus prioritaires que d'autres.

Système de règles par défaut. Un participant aux pré-tests a exprimé l'idée d'une « *ligne de base* » consistant en un ensemble de phrases fournies au départ. Ces règles devraient définir un comportement de confort « *minimum* », tel que l'allumage des lampes en fonction des capteurs par exemple. Un autre participant (S5) a utilisé cette notion en l'appliquant directement au scénario : pour lui, l'utilisateur va programmer deux scénarios pour sa routine matinale. Le scénario de base, avec la gestion de lumières automatiques par exemple, et le scénario tout confort, qui correspond au scénario de base avec toutes les briques nécessaires à la notion de confort de l'utilisateur. « *Le scénario tout confort s'exécutera dans 80% des cas, quand tout se passera bien, quand l'utilisateur sera à l'heure, et qu'il n'y aura pas de tempête dehors, etc. Le scénario de base sera, lui, utilisé dans les 20% de cas restants* ».

Nous retrouvons ici l'importance de l'exception dans les activités routinières.

Utilisation d'autres langages. Au cours des tests, il est apparu que le LPN n'était pas toujours bien adapté. Par exemple, la définition de conditions horaires pourrait s'appuyer sur des lignes de temps [25]. L'approche multisyntaxe avec égale opportunité totale est clairement souhaitable. KISS, qui intègre un langage abstrait pivot, ouvre cette possibilité, mais seule la version LPN est actuellement opérationnelle.

CONCLUSION ET PERSPECTIVES

Nous avons présenté KISS, un outil de développement par l'utilisateur final pour l'habitat intelligent. L'évaluation expérimentale montre que les participants ont pu programmer avec succès le scénario imposé. L'utilisation du langage pseudonaturel et l'utilité d'un habitat virtuel

dual de l'habitat réel ont été comprises. Les difficultés rencontrées portent pour l'essentiel sur le vocabulaire limité et sur l'interprétation de certaines phrases incluant des verbes progressifs. Pour répondre à ces lacunes, nous posons comme perspectives à ce travail la co-construction d'un langage multisyntaxe entre l'utilisateur et le système. Cette co-construction pourrait s'appuyer sur les techniques de programmation par et avec exemples [15]. Le passage à l'échelle et le multi-utilisateur doivent d'être considérés.

REMERCIEMENTS

Cette recherche a été soutenue par le projet ANR-08-VERS-005 CONTINUUM (CONTinuité de service en INformatique UbiqUitaire et Mobile).

REFERENCES

1 Ash, J. et al. Scratchable devices: user-friendly programming for household appliances. In *Proc. of the 14th int. conf. on Human-Computer Interaction*, Vol. 3 (HCII'11), 2011, 137-146

2 Barraquand, R. *Designing sociable technologies*, thèse de l'université de Grenoble, 2012

3 Bell, G. et al. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. *Personal Ubiquitous Computing*, 11, 2 (2007), 133-143

4 Blackwell, A. et al. AutoHan: an architecture for programming the home. *In Proc. of the IEEE Symposia on Human Centric Computing Languages and Environments*, 2001, 150-157

5 Blackwell, A. et al. First steps in programming : a rationale for attention investment models. In Proc. *of the IEEE Symposia on Human Centric Computing Languages and Environments*, 2-10, 2002

6 Coutaz, J. et al. DisQo : A user needs analysis method for smart home. In *Proc. NordiCHI'10*, ACM, 615-618

7 Dey, A. K., et al. a CAPpella: programming by demonstration of context-aware applications. In *Proc. of CHI'04*. ACM, 33-40

8 Dey, A. K. et al. iCAP: interactive prototyping of context-aware applications. In *Proc. Pervasive'06*, Springer, 254-271

9 Dix, A. et al. *Human Computer Interaction*, Prentice Hall, 2004

10 Dixon, C., et al. The home needs an operating system (and an app store). In *Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (Hotnets '10). ACM, New York, NY, USA, Article 18

11 Drey, Z. et al. A Visual, Open-Ended Approach to Prototyping Ubiquitous Computing Applications. In *Proceedings of PERCOM 2010*

12 Fisher, M. et al. Integrating automated test generation into WYSIWYT spreadsheet testing methodology. In *ACM Trans. on Software Engineering Methodology (TOSEM)*, 15, 2, 150-194

13 Fontaine, E. *Programmation d'espace intelligent par l'utilisateur final*. Thèse Université de Grenoble, 2012

14 Garcia-Herranz, et al. Towards a Ubiquitous End-User Programming System for Smart Spaces. *Journal of Universal Computer Science*, 16(12), 012, 1633-1649

15 Gil, Y. et al. TellMe: learning procedures from tutorial instruction. In *Proc.* IUI'11, 227-236

16 Girard, P. *Ingénierie des systèmes interactifs : vers des méthodes formelles intégrant l'utilisateur*. Thèse HDR, Université de Poitiers, 2000.

17 Jouve, W., et al. DiaSim: A parameterized simulator for pervasive computing applications. In *Proc. of PERCOM '09*. IEEE, Washington DC, 1-3

18 Ko, A et al. B. Designing the Whyline: a debugging interface for asking questions about program behavior. In *Proc. of CHI'04*, 2004, 151-158

19 Ko, A. J. et al. The State of the Art in End-User Software Engineering. *In ACM Computing Surveys*, 43 (3), 2011, 1-44

20 Lieberman, H., et al. End-user development: An emerging paradigm. In *End User Development (Human-Computer Interaction Series)*. 2006, 1-8

21 Maloney, J. et al. The Scratch Programming Language and Environment. *Trans. Computer Education*, 10(4), 2010

22 Merrill, D. et al. Siftables : towards sensor network user interfaces. In *Proc. TEI'07*, ACM, 2007, 75-78

23 Nardi, B. *A Small Matter of Programming, Perspectives on End User Programming*. MIT Press, 1993

24 Paille, P et al, A. *L'analyse qualitative en sciences humaines et sociales*, Colin, 2008. 312 pages

25 Plaisant, C. et al. Scheduling on-off home control devices. In *Proc.CHI '91*, ACM, 459-460

26 Rogers, Y. Moving on from Weiser's vision of calm computing: engaging ubicomp experiences. In *Proc. of UbiComp'06*, 404-421

27 Sousa, J. P. et al. TeC: end-user development of software systems for smart spaces. *Int. J. Space-Based and Situation Computing*, 1(4), 2011

28 Tennant, H. Menu-based natural language understanding. In *Proc. of national computer conference and exposition (AFIPS '84)*. ACM, 629-635

29 Truong, K., et al. Camp: A magnetic poetry interface for end-user programming of capture applications for the home. *UbiComp'04*, 2004, 140-160