HAL
open science

Programmation d'espace intelligent par l'utilisateur final

Emeric Fontaine

To cite this version:

Emeric Fontaine. Programmation d'espace intelligent par l'utilisateur final. Interface homme-machine [cs.HC]. Université de Grenoble, 2012. Français. NNT : 2012GRENM034 . tel-00744415

HAL Id: tel-00744415

https://theses.hal.science/tel-00744415v1

Submitted on 23 Oct 2012

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.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

« Emeric FONTAINE »

Thèse dirigée par **« Joëlle COUTAZ »**
et codirigée par **« Alexandre DEMEURE »**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
Équipe Ingénierie de l'Interaction Homme-Machine
dans **l'École Doctorale *Mathématiques, Sciences et Technologies de l'Information, Informatique***

Programmation d'espace intelligent par l'utilisateur final

Thèse soutenue publiquement le **« date de soutenance »**,
devant le jury composé de :

Mme., Regina , Bernhaupt
Professeure invitée, Université Paul Sabatier, Rapporteur
M., Patrick, Girard
Professeur, Université de Poitiers, Rapporteur
Mme., Marie-Christine, ROUSSET
Professeure, Université Joseph Fourier, Présidente
M., Jean-Yves, Tigli
Maître de conférences, Université Nice Sophia Antipolis, Examinateur

Comtois, *rends-toi ! Nenni, ma foi !*

Remerciements

Ces remerciements sont pour vous, lecteurs du préambule à ce mémoire. Cependant ils représentent pour moi, l'épilogue de ce travail doctoral. Cette thèse doit beaucoup aux différentes personnes que j'ai côtoyées durant ces trois dernières années. Mon inquiétude désormais est de n'oublier personne.

Mes premiers remerciements vont à Joëlle Coutaz et Alexandre Demeure, pour m'avoir fait l'honneur d'être le relais entre le dernier thésard de l'une et le premier de l'autre. Ces remerciements leur vont aussi pour un grand nombre d'autres raisons qui ont abouti à la qualité de ce travail de recherche doctorale. Cette qualité à été apréciée par mon jury de thèse que je souhaite également remercier. A commencer par mes rapporteurs, Regina Bernhaupt et Patrick Girard ; puis mes examinateurs, Marie-Christine Rousset et Jean-Yves Tigli, et pour finir, par Jean Caelen qui a répondu présent à mon invitation.

Ces travaux auraient une saveur bien différente sans l'aide de Nadine Mandran, que je remercie de m'avoir appris à monter une étude, mener des entretiens auprès d'utilisateurs ainsi que pour les échanges de confitures.

Je remercie tous les membres de l'équipe IIHM passée et présente, en particulier ceux qui ont eu à me supporter au quotidien (les chanceux) Audrey, Fred et Yoann qui s'en sont allés puis Celine et Laure qui doivent (je l'imagine) s'ennuyer suite à mon départ. Cette équipe recelle également un fort potentiel d'acteurs malheureusement méconnus : Fatou, William et Mathieu. Je leur souhaite une longue carrière et encore merci à eux.

Je souhaite remercier les différentes personnes avec qui j'ai eu la chance de travailler, je pense tout d'abord aux Niçois, en particulier, Vincent, Stéphane, Gaëtan et Anis (si si, il est Niçois maintenant). Puis aux membres de l'équipe MultiCom du LIG pour leur aide lors de la réalisation de l'évaluation de KISS : Sybille, Mathieu et Nicolas. Enfin je pense à Mohamad et Jonathan pour leur contribution au développement de l'environnement virtuel. Ce fut un véritable plaisir que de travailler avec eux.

Je me dois de remercier les 17 foyers qui m'ont ouvert leurs portes pour mon étude de terrain, ainsi que les 13 participants à l'évaluation, sans oublier les 3 participants pilotes qui ont beaucoup souffert (les pauvres).

Mes derniers remerciements vont à mes proches, le RoroGroup et ma famille pour leur présence et leur soutien ; et enfin à Christelle pour sa patience au quotidien.

Tables des matières

Chapitre I : Introduction 9

Table des matières : 9
I – Sujet et motivation 10
II – Définitions, intelligence ambiante et espaces intelligents 10
II-1 Intelligence ambiante 10
II-2 Espace intelligent 12
III – Constats et hypothèses 13
IV – Objectifs de la thèse et verrous traités 15
V – Approche 17
VI – Portée et limite de l'étude 18
VII – Organisation du mémoire et résumé des contributions 18

Chapitre II : Etude de terrain 21

Table des matières : 21
Résumé : 21
Avant-propos 22
I – Méthodes d'analyse des besoins pour l'habitat 23
I-1 Mesures isolées 23
I-2 Mesures combinées 24
II – Le protocole expérimental 25
II-1 Situer l'étude par rapport au contexte sociétal 25
II-2 DisQo : méthode d'analyse de la perception et des attentes des ménages vis-à-vis de l'habitat intelligent 28
II-3 Recrutement des participants 30
III – Résultats de l'étude DisQo 31
III-1 Confirmation de résultats connus 31
III-1-1 Moments clefs et temporalité 32
III-1-2 Classes de services pour une meilleure qualité de vie 33
III-1-3 Réticences 36
III-2 Résultats originaux 37
III-2-1 Emergence de sens et capacités des objets assemblés 37
III-2-2 Nouvelles fonctions attendues des assemblages d'objets et algèbre d'assemblage 39
IV – Conclusion 41

Chapitre III : Utilisateur final, programmation et génie logiciel : espace problème et état de l'art 42

Table des matières 42
Résumé 43
Avant-propos 45
I – PUF-DUF-GLUF : définitions et importance 45
I-1 PUF - Programmation par l'Utilisateur Final 45
I-2 Développement par l'Utilisateur Final et Génie Logiciel par l'Utilisateur Final 46
I-3 Motivations pour des outils PUF-DUF-GLUF 48
II – Espace problème PUF-DUF-GLUF 49
II-1 Classification de l'utilisateur final 50
II-2 Caractérisation de l'environnement de développement et d'exécution 52
II-2-1 Couverture fonctionnelle 52
II-2-2 Support au co-développement 53
II-2-3 Couplage entre phase de conception/développement et phase d'exécution 54

II-3 Caractérisation des langages de développement et outils d'édition 55
II-3-1 Paradigmes de programmation 55
II-3-2 Spécificité contre Généralité 56
II-3-3 Extensibilité 56
II-3-4 Expressivité et adaptation 57
II-3-5 Lisibilité et facilité d'écriture 58
III – Exemples représentatifs de l'Etat de l'art 61
III-1 a CAPpella 61
III-2 iCAP 63
III-3 CAMP 66
III-4 AutoHAN 67
III-5 BYOB appliqué aux dispositifs domestiques 70
III-6 TeC 71
III-7 PANTAGRUEL 75
IV – Analyse synthétique de l'état de l'art 78

Chapitre IV : KISS, un environnement de développement par l'utilisateur final, et son infrastructure d'exécution, CONTINUUM 84

Table des matières : 84
Résumé :..... 85
Avant-propos 86
I – CONTINUUM, l'infrastructure d'exécution 87
I-1 Principes directeurs de WCOMP 87
I-2 Architecture logicielle de CONTINUUM 92
II – Un exemple de méta-IHM réflexe 98
II-1 L'interaction dans PhotoBrowser 99
II-2 Mise en œuvre 100
III – KISS vu de l'utilisateur final 102
III-1. KISS dans l'espace problème PUF-DUF-GLUF 102
III-2 KISS en action 103
IV – Mise en œuvre de KISS..... 108
IV-1 Présentation générale 108
IV-2 Grammaire KISS du langage de programmation 110
IV-3 Analyseur LX-SX et la représentation pivot 112
IV-4 L'éditeur de programme EdProg 115
IV-5 Analyseur sémantique et génération de code 119
IV-6 Simulateur, metteur au point 129
V – Conclusion 130

Chapitre V : L'acceptabilité de KISS 132

Table des matières 132
Résumé 132
Avant-propos 134
I – Préparatifs expérimentaux 135
I-1 Influence du scénario 136
I-1-1 Adaptation de l'extrait de scénario 136
I-1-2 Ajout d'équipements dans DOMUS 137
I-1-3 Utilisation de la technique du Magicien d'Oz 137
I-2 Influence de l'infrastructure d'accueil 139
I-2-1 Extension logicielle des équipements UPnP 139
I-2-2 Remplacement de WCOMP par un intergiciel dédié 139
I-2-3 Simulation de la Base de Connaissances (BdC) 141
I-2-4 L'habitat virtuel et le metteur au point 141
II – Protocole expérimental 142

II-1 Déroulement de l'expérimentation 143
II-2 Recrutement des participants 145
III - Résultats du test d'acceptabilité 145
III-1 Résultats relatifs à notre approche 146
III-1-1 Aspects positifs 146
III-1-2 Difficultés rencontrées 148
III-1-3 Pistes d'améliorations 151
III-2 Résultats relatifs à KISS 152
III-2-1 Aspects positifs 152
III-2-2 Difficultés rencontrées 153
III-2-3 Pistes d'améliorations 153
IV – Conclusion 155

Chapitre VI : Conclusion 156

Table des matières : 156
I - L'apport de la thèse 157
II – Du laboratoire au terrain : nos recommandations 162
III – Considérations éthiques 164

Bibliographie 166

Annexes 173

Table des matières : 173
ANNEXE II-1 : Extrait du rapport du CREDOC 174
ANNEXE II-2 : Grille d'entretien auprès des industriels 182
ANNEXE II-3 : Scénario directeur pour le développement de KISS 186
ANNEXE IV-1 : Grammaire de KISS 191
ANNEXE V-1 : Déroulement de l'expérimentation 195
ANNEXE V-2 : Questionnaire d'évaluation 199
ANNEXE V-3 : KISS pour les participants et le magicien d'Oz 201

Chapitre I : Introduction

Table des matières :

I – Sujet et motivation 10
II – Définitions, intelligence ambiante et espaces intelligents 10
III – Constats et hypothèses 13
IV – Objectifs de la thèse et verrous traités 15
V – Approche 17
VI – Portée et limite de l'étude 18
VII – Organisation du mémoire et résumé des contributions 18

I – Sujet et motivation

> Nos travaux de recherche doctorale traitent du problème de la construction d'espaces intelligents sous l'angle de l'Interaction Homme-Machine (IHM).

Le domaine de l'IHM a pour mission la conception et la réalisation de systèmes interactifs utiles, utilisables, en contexte – utiles, c'est-à-dire en conformité avec les services attendus par l'utilisateur cible ; utilisables, c'est-à-dire conformes aux capacités sensori-motrices et cognitives de cet utilisateur ; en contexte, car adaptés à la situation.

Dans la pratique actuelle, la conception d'un système est assurée par des spécialistes (les designers), la réalisation revient à d'autres spécialistes (les informaticiens) et l'utilisateur est assimilé à une représentation archétypique réductrice (cf. le concept de persona (Carroll 2000) ou (Preece et al. 2002)). Il en résulte une double dichotomie : d'une part, la séparation entre phase de conception et phase d'exécution, d'autre part, la distinction entre développeurs et utilisateurs. Au final, l'utilisateur, en bout de chaîne du processus de production, est un consommateur contraint par un système pensé et réalisé par d'autres.

> Dans cette thèse, nous prenons le contre-pied de cette pratique en proposant de redonner le pouvoir à l'utilisateur final. Dans cet esprit, l'utilisateur doit pouvoir tenir les rôles de concepteur et de développeur pour construire à façon un espace intelligent dans lequel les phases de conception et d'exécution sont unifiées en un tout cohérent.

L'intelligence ambiante et notamment les espaces intelligents, en pleine émergence, constituent un terrain propice à la mise en pratique de ce point de vue. Que recouvrent ces termes ?

II – Définitions, intelligence ambiante et espaces intelligents

II-1 Intelligence ambiante

L'intelligence ambiante trouve son origine dans la révolution imaginée par Mark Weiser il y a une vingtaine d'années, qu'il nomme « informatique ubiquitaire[1] » (Weiser 1991). Weiser entrevoit une informatique du futur ancrée dans nos activités quotidiennes au point de s'y fondre jusqu'à disparaître. Il illustre son propos avec l'exemple de l'écriture : omniprésente dans nos sociétés modernes, chacun l'utilise au quotidien sans même y prêter attention. Le terme « informatique ubiquitaire » s'est vu décliner ensuite en « informatique diffuse[2]» – qui se répand progressivement et

1 Traduction de « ubiquitous computing ».
[2] Traduction de « pervasive computing».

inexorablement, puis en « intelligence ambiante » – qui environne de manière permanente.

> La finalité de l'intelligence ambiante est de « fournir des espaces de services et des dispositifs techniques capables de répondre de manière adaptée en toute circonstance à la fois à des besoins individuels et à des défis sociétaux dans tous les secteurs d'activités » *(Coutaz et al. 2012).*

Par exemple, favoriser des modes de vie plus économes en consommation énergétique tout en créant des cadres de vie confortables et attractifs, renforcer une autonomie heureuse des personnes âgées ainsi que la cohésion des familles face aux nouveaux modes de vie, ou encore soutenir la participation sociale pour la création de richesses intellectuelles et matérielles.

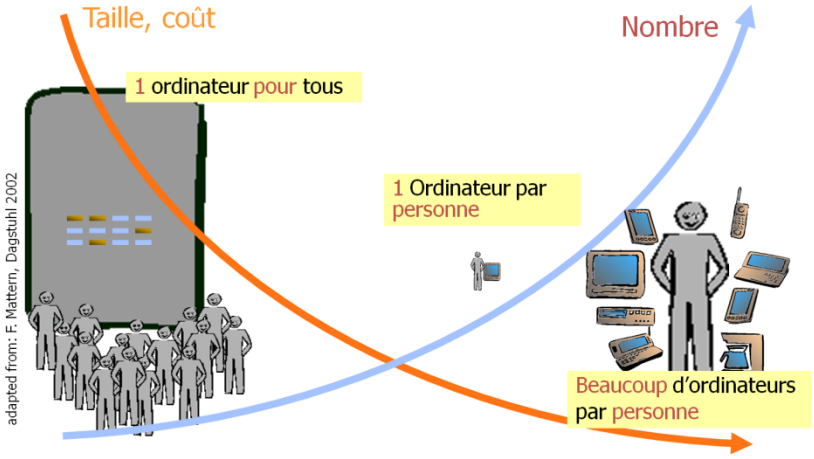


Figure I-1. Conséquence des progrès et convergence dans le domaine des réseaux, de la micro-électronique, et du logiciel. [adapté de (Mattern 2002)]

Cette vision de l'intelligence ambiante est rendue possible par la convergence des progrès technologiques : miniaturisation continue et puissance grandissante des composants électroniques, chute des coûts de production, déploiement à grande échelle des réseaux et moyens de communication sans fil. Cette convergence a pour conséquence l'augmentation spectaculaire du nombre d'équipements électroniques par personne. La figure I-1 (Mattern 2002) traduit cette évolution selon laquelle nous passons d'un ordinateur pour tous dans les années soixante, à un ordinateur par personne dans les années quatre-vingt, pour atteindre aujourd'hui de nombreux processeurs par individu.

De plus, les ordinateurs ne se limitent plus à l'unité centrale, l'écran, la souris et le clavier usuels, mais à l'augmentation des équipements du quotidien par des services numériques et à leur interconnexion. Le boîtier Internet, qui permet de relier téléviseur, téléphone fixe et équipements sans fil comme l'ordinateur et le téléphone cellulaire, illustre cette tendance. D'autres équipements et services interconnectables sont déjà disponibles sur le marché comme le Nabaztag, la LiveRadio d'Orange, ou l'aspirateur Roomba de iRobot (Figure I-2). On parle alors de "domicile réticulé"[3]

[3] En anglais, "networked home".

(Hague et al. 2006). La mise en réseau robuste et dynamique de dispositifs hétérogènes est une condition nécessaire à l'enrichissement numérique du monde physique pour le déploiement d'espaces (ou environnements) "intelligents".

Figure I-2. Equipements du marché actuel. De gauche à droite, le Nabaztag, la LiveRadio, le Roomba.

II-2 Espace intelligent

Nous convenons de définir un espace intelligent comme suit.

> Un espace (ou environnement) intelligent est un lieu (physique ou virtuel) d'activités (d'origine humaine ou non) constitué de dispositifs et de fonctions interconnectés, capable de garantir en toute circonstance les services attendus avec la valeur attendue.

Expliquons les termes clefs de cette définition.

- *Intelligent* implique adéquation, adaptation et respect de la valeur. Il n'implique pas, sans toutefois l'exclure, l'utilisation des paradigmes de l'Intelligence Artificielle pour la mise en œuvre technique.
- La *valeur* peut s'exprimer de manière réductrice en termes de qualité (au sens du Génie Logiciel). Mais comme le fait remarquer Cockton (Cockton 2004) et (Cockton 2005) puis Calvary (Calvary 2007), il s'agit d'aller au-delà de métriques avec la prise en compte notamment de la satisfaction, du contentement, du plaisir, de toute sensation relevant du bien-être.
- Un *service* désigne une assistance ou commodité immatérielle fournie par l'espace intelligent. Il n'implique pas, sans toutefois l'exclure, l'utilisation du paradigme "Service-Oriented Architecture" pour la mise en œuvre technique.
- L'espace intelligent n'est *pas nécessairement un lieu clos* (l'activité en question peut être menée dans la rue, en pleine nature) ; il n'est *pas nécessairement physique* (l'activité peut avoir lieu dans un monde virtuel comme dans Second Life[4]). En vérité, il est une réalité mixte où le physique et le numérique se fondent en une "nouvelle réalité".
- Dès lors, les *dispositifs* auxquels nous faisons référence sont des artefacts (c.-à-d. des créations humaines) pouvant jouir d'*une double existence* – dans le monde physique et/ou, comme notre thèse le propose, dans un monde dual numérique.

[4] http://secondlife.com/

Notre définition assimile un espace intelligent à un lieu d'activités. En cela, nous reprenons à notre compte l'analyse de Harrison et de Dourish (Harrison et al. 1996) qui écrivent ceci : "Nous sommes situés dans un espace, mais nous agissons en des lieux. En outre, un lieu est un espace auquel nous attachons une valeur. C'est ce qui fait la différence entre une maison et un domicile. La maison nous protège du vent et de la pluie, mais le domicile, c'est l'endroit où nous vivons."[5]. Alors que Harrison et Dourish font référence à l'espace physique uniquement, nous étendons leur analyse aux espaces numériques. En effet, le concept d'espace de travail, introduit pour la génération d'Interfaces Homme-Machine, désigne une entité de structuration de ces interfaces qui réunit un ensemble de tâches (donc, d'activités) logiquement connectées (Normand 1992). Ainsi, un lieu d'activités peut être aussi de nature numérique.

Dans la suite de ce mémoire, nous conviendrons qu'un *habitat intelligent* est un cas particulier d'espace intelligent.

> Un habitat intelligent est un lieu d'activités (d'origine humaine ou non) constitué de dispositifs et de fonctions interconnectés, capable de garantir en toute circonstance les services attendus par ses habitants avec la valeur attendue comme le confort, l'économie d'énergie, la sécurité des biens et des personnes et plus généralement, le bien-être.

III – Constats et hypothèses

À l'heure actuelle, les espaces et habitats intelligents font l'objet de nombreuses recherches dans tous les domaines fondateurs de l'intelligence ambiante, mais souvent de manière cloisonnée (micro-électronique, réseau et informatique notamment). En tant que spécialité de l'Informatique, l'Interaction Homme-Machine y a sa part avec la conception de nouvelles techniques d'interaction (Mistry 2009) (Harrison et al. 2010) et (Harrison et al. 2011), la plasticité des interfaces (Calvary 2003) et (Calvary 2007) ou encore l'étude de technologies persuasives (Fogg 2003). Toutefois, comme nous le verrons au chapitre III sur l'état de l'art, peu de travaux en IHM se sont penchés sur le problème du contrôle laissé à l'utilisateur (Humble et al. 2003) (A. Dey et al. 2006) et (Newman et al. 2008). Il nous paraît urgent d'étudier cette question au regard de la recherche en système réparti et intergiciel qui, elle, vise l'autonomie des systèmes sans « utilisateur dans la boucle ». Nous proposons d'étudier ce problème du contrôle non seulement à l'exécution du système, mais en amont dès la conception en sorte que l'utilisateur final ne soit plus confiné au seul rôle de consommateur mais puisse être le bâtisseur de ses propres espaces intelligents.

[5] "We are *located* in "space", but we *act* in "place". Furthermore, "places" are spaces that are valued. The distinction is rather like that between a "house" and a "home"; a house might keep out the wind and the rain, but a home is where we live. "

Cette vision du futur s'appuie sur les deux hypothèses suivantes : (1) l'utilisateur a le potentiel d'être un créateur d'espace intelligent ; (2) un espace intelligent se construit de manière incrémentale par assemblage de dispositifs et de services.

1ère hypothèse : l'utilisateur est un créateur d'espace intelligent

Concernant la première hypothèse, les chercheurs en psychologie cognitive attestent que l'être humain est naturellement créatif (Amabile 1983). *"The general psychological conviction seems to be that all individuals possess to some degree all abilities [...] Creative acts can therefore be expected [...] of almost all individuals."* (Guilford 1987). Le Do It Yourself (DIT), c'est-à-dire *«[...] toute activité où l'on n'est plus spectateur ou consommateur [...] »* [6], en est l'illustration. Le succès du DIT a conduit le marché à offrir des solutions en kit (typiquement des meubles) en sorte que des activités autrefois réservées aux professionnels ou aux bricoleurs experts, sont aujourd'hui accessibles à tous.

Figure I-3. DataTiles de Rekimoto (Rekimoto et al. 2001). Les DataTiles sont des tuiles en plastique transparent. Chaque tuile a un nom et un service associé. Lorsqu'une tuile est posée sur l'écran, le service s'exécute et l'interaction avec ce service est possible à travers la tuile. Les tuiles sont assemblées selon un modèle flux de données pour fournir de nouveaux services. Par exemple, sur cette image, un service « météo » est couplé à un service « machine à parcourir le temps » ce qui permet de visualiser le temps qu'il a fait ou qu'il fera.

2ème hypothèse : un espace intelligent se construit par incréments

Par analogie et par extension, nous pensons que l'utilisateur final est capable de construire des espaces intelligents par assemblage et programmation de dispositifs et de services, sous réserve que les briques élémentaires soient, comme pour les kits, bien conçues. Des travaux précurseurs accréditent cette hypothèse : les Lego-Logo et les Lego Mindstorms fondés sur les théories de Seymour Papert[7] permettent, par assemblage et programmation, de construire des objets robotisés. Les Triangles de Ishii peuvent être assemblés pour élaborer des contes (Gorbet et al. 1998) et, chez Sony, les Data Tiles (Rekimoto et al. 2001) permettent de créer des services à façon (Figure I-3). Plus récent, le LilyPad Arduino, alliant art et technologie, facilite, entre autres, la réalisation de nos propres e-vêtements (par exemple, des flèches lumineuses « brodées » au moyen de LED au dos d'une veste pour indiquer un changement de direction) (Buechley et al. 2008). À l'instar des logiciels libres, nous aurons demain du

[6] http://fr.wikipedia.org/wiki/Do_it_yourself

[7] Seymour Papert : mathématicien et pionnier de l'Intelligence Artificielle au MIT.

« matériel libre » disponible de manière collective. La plate-forme BUG de Bug labs illustre cette tendance[8]. À terme, il est envisageable de concevoir des infrastructures de conception collective avec et par les utilisateurs.

Ces constats et hypothèses motivent les objectifs suivants.

IV – Objectifs de la thèse et verrous traités

Les objectifs de cette thèse comprennent trois volets :

Objectifs en 3 volets

(1) Vérifier la validité de nos hypothèses : que l'utilisateur final souhaite avoir la possibilité de devenir le concepteur et le développeur de ses propres lieux de vie et qu'un espace intelligent se construit par assemblage de dispositifs et de services. Si cette hypothèse est vérifiée, alors :

(2) Fournir à l'utilisateur final des outils de type « Programmation par l'Utilisateur Final » - « Développement par l'Utilisateur Final » (PUF-DUF)[9] en sorte qu'il puisse construire facilement et de manière opportuniste des espaces intelligents. Les outils PUF-DUF se doivent de couvrir tout ou partie des activités de développement (programmation, test, maintenance et réutilisation, etc.), mais s'adressent au non-professionnel. A priori, ce non-professionnel n'a pas pour objectif d'apprendre les principes du Génie Logiciel, mais de construire un nouvel objet, dans notre cas, un espace intelligent.

(3) Valider le bien fondé des outils PUF-DUF par l'expérimentation.

Trois verrous

Ces objectifs doivent permettre de lever trois verrous : la question de la couverture fonctionnelle des outils PUF-DUF et plus spécifiquement la puissance d'expression du langage de programmation ; le problème de la cohabitation de l'interaction explicite et de l'interaction implicite ; l'équilibre entre l'autonomie d'un espace intelligent et le contrôle laissé à l'utilisateur.

- Couverture fonctionnelle des outils PUF-DUF et puissance d'expression du langage de programmation. Tous les outils de création actuels, depuis les Lego-Logo jusqu'au kit Arduino, impliquent une activité de type « programmation par l'utilisateur final». L'expérience montre qu'ils reçoivent un franc succès. Toutefois, le pouvoir d'expression de ces outils est ajusté à la construction de gadgets simples, non pas à la programmation d'objets complexes comme les espaces intelligents. Aujourd'hui, la « programmation » des équipements du domicile relève du paramétrage dont la couverture fonctionnelle est fixée par le fabricant. Nous verrons dans l'état de l'art du chapitre III que les outils PUF-DUF pour espaces intelligents vont bien au-delà du gadget et du paramétrage. Toutefois, ils mettent l'accent sur les activités de

[8] La plate-forme BUG (www.buglabs.net) permet à l'utilisateur de se fabriquer des objets à partir de modules électroniques, par exemple un appareil photo qui note la position géographique de la prise de vue sur une carte pour l'envoyer ensuite sur Flickr, site de référence en matière de partage de photos. Vidéo de démonstration : www.youtube.com/watch?v=N5May_FzcvE

[9] Traduction de l'anglais « end-user programming » - « end-user development » (EUP/EUD).

programmation, laissant au second plan la mise au point, la réutilisation, le déploiement et l'administration (Edwards et al. 2001).

> Dans cette thèse, la mise au point est une composante clef de l'espace des outils à fournir tandis que les aspects déploiement et administration sont déportés sur les capacités de l'infrastructure logicielle d'accueil.

- Cohabitation de l'interaction explicite et de l'interaction implicite. L'intelligence ambiante pose le problème aigu de l'*interaction incidente* ou *implicite* "dans laquelle des actions [humaines] sont réalisées pour d'autre objectif [que l'effet résultant de ces actions] ou des signes [par exemple physiologiques] sont produits inconsciemment et interprétés [par le système] en vue d'influencer/améliorer/faciliter l'interaction future des acteurs ou leur quotidien" (Dix 2002). Prenons un exemple. L'action humaine « entrer dans une pièce » dans le but de récupérer un objet oublié (objectif explicite) a pour effet d'allumer une lampe de la pièce (interprétation système de l'action humaine). Il s'agit d'une interaction incidente (implicite) puisque l'action de l'utilisateur (entrer) n'a pas pour objectif d'éclairer la pièce. Nous aurions assisté à une interaction explicite si l'utilisateur avait eu pour objectif d'allumer la lampe et s'il avait choisi d'atteindre cet objectif en pénétrant dans la pièce (au lieu d'utiliser un interrupteur).

> Nous considérons que ce glissement dynamique entre interaction explicite et interaction implicite/incidente peut être résolu si l'utilisateur a (ou a eu) la possibilité de spécifier l'effet d'une interaction implicite, par exemple « quand je rentre dans cette pièce et qu'il fait noir, allumer la lampe».

La bonne cohabitation entre interaction explicite et implicite est donc, de notre point de vue, liée au niveau de contrôle laissé à l'utilisateur.

- Niveau de contrôle laissé à l'utilisateur et autonomie des espaces intelligents. Puisque l'espace intelligent est, par définition, doué de facultés d'adaptation, il convient de s'interroger sur son degré d'autonomie et de poser la question duale du niveau de contrôle laissé à l'utilisateur. Dans l'exemple précédent, l'effet incident (allumer la lampe) provoqué par notre utilisateur aurait pu gêner une autre personne en train de dormir dans la pièce en question. Si l'éclairage est câblé par les concepteurs sur la seule détection de l'entrée d'une personne, nous sommes en droit d'affirmer que l'espace n'est pas si intelligent que ça ! Aujourd'hui, une adaptation autonome bien conçue repose sur l'interprétation et l'implémentation, par les concepteurs et par les développeurs, des requis (pas toujours clairement exprimés) de l'utilisateur. Or ces requis varient de manière opportuniste : après tout, l'utilisateur de notre exemple vivait seul au moment de la commande ou de l'achat du système. Le Génie Logiciel (GL) s'intéresse à la prise en compte des requis dynamiques, mais conserve la séparation entre les phases de conception et d'exécution (séparation que nous souhaitons gommer dans cette thèse). Avec l'approche par composants orientés services, le GL propose l'administration, la reconfiguration et le déploiement dynamiques, de même l'évolutivité des systèmes (Ferry et al. 2011) (Escoffier et al. 2007).

> Dans cette thèse, nous nous appuyons sur une infrastructure orientée service, mais au sein de laquelle nous définissons des points de contrôle ouverts sur l'utilisateur.

V – Approche

Pour atteindre nos objectifs et répondre aux verrous posés, nous avons adopté une approche expérimentale guidée par des scénarios d'usage (voir figure I-4). Ces scénarios présentés en annexe II-3 couvrent certaines activités routinières du quotidien (le lever) ainsi qu'une séquence vacances au restaurant.

Nos scénarios d'usage reflètent l'étude de terrain préliminaire que nous avons conduite auprès de dix-sept familles de la région grenobloise. Cette étude devait nous permettre de valider notre hypothèse : l'utilisateur final est-il prêt à réaliser des assemblages de dispositifs et de services pour améliorer sa qualité de vie ?

La réponse nous étant favorable, nous avons conçu et développé le PUF-DUF KISS (Knit your Ideas into Smart Spaces) en tenant compte de l'état de l'art et en nous servant de CONTINUUM[10], une infrastructure d'accueil orientée services, dans laquelle nous avons pris soin de définir les points de contrôle nécessaires à l'utilisateur.

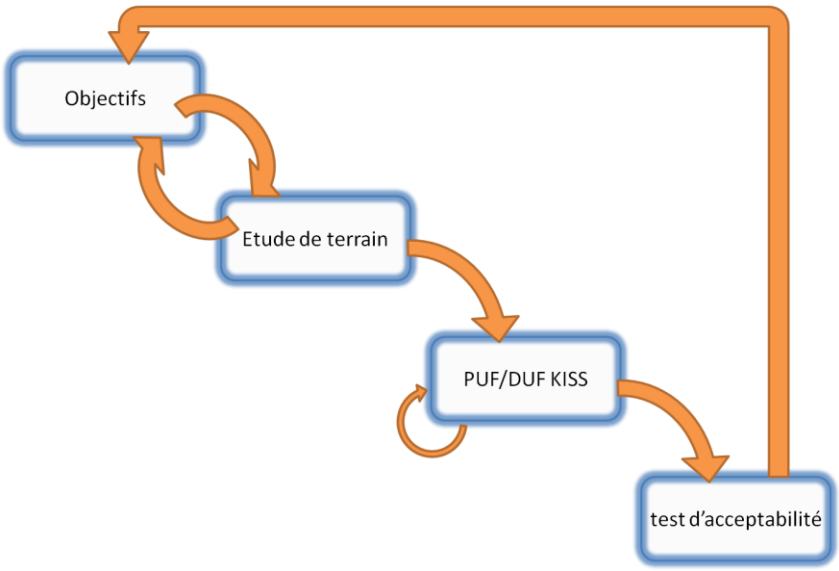


Figure I-4. Les étapes de notre approche

KISS a fait ensuite l'objet de test d'acceptabilité avec 13 utilisateurs représentatifs en situation semi-contrôlée au moyen de la plate-forme DOMUS du laboratoire LIG. DOMUS reproduit un appartement entièrement meublé et équipé de capteurs et de dispositifs contrôlables (lumière, chauffage, volets roulants) mais DOMUS ne permet pas d'y vivre plusieurs jours de suite (d'où la situation semi-contrôlée).

[10] https://continuum.unice.fr/

VI – Portée et limite de l'étude

Notre étude porte sur le développement par l'utilisateur final d'habitats intelligents. Il s'agit d'un problème complexe que nous traitons sous le seul angle de l'Interaction Homme-Machine. Autrement dit, les contributions visées ne se situent pas dans le domaine des systèmes répartis, des systèmes critiques ou de l'intelligence artificielle, mais interviennent en synergie avec ces domaines.

Si notre recherche relève de l'Interaction Homme-Machine, les contributions visées s'inscrivent dans le courant de recherche dit « IHM systémique » (par complémentarité avec la recherche « IHM à pointe fine ») (Coutaz et al. 2012) :

- L'*IHM à pointe fine* est centrée sur l'invention de nouvelles techniques d'interaction pour résoudre des micro-problèmes d'interaction, par exemple, le pointage de cible dans une scène 2D.
- L'*IHM systémique* observe une approche holistique selon laquelle le système est envisagé comme un tout pour des usages dans le monde réel avec sa diversité et ses aléas. Les méthodes de conception, les architectures logicielles de systèmes interactifs, les boîtes à outils et les générateurs d'IHM relèvent de ce courant.

Ainsi, notre étude sur le développement par l'utilisateur final relève de l'IHM systémique. Elle n'exclut pas l'invention de nouvelles techniques d'interaction pour programmer et mettre au point les programmes dans un habitat intelligent, mais ce n'est pas notre visée première. Nous visons la création d'outils adaptés et leur intégration dans un environnement technique équipé « intelligence ambiante ». En raison de la complexité du sujet, cet environnement technique est supposé fermé, par opposition aux milieux ouverts en plein air. De même, bien qu'il s'agisse d'habitat intelligent, d'habitants et de milieu familial, notre recherche étudiera le multi-utilisateur au niveau de l'état de l'art seulement, mais se limitera à une utilisation mono-utilisateur.

> En synthèse, cette étude porte sur le problème de la programmation et de la mise au point de programmes par l'utilisateur final. En raison de la complexité du sujet, nous nous limitons à une étude centrée IHM systémique pour mono-utilisateur, applicable en milieu fermé dans un habitat intelligent, sans aborder les problèmes de protection de l'espace privé ni ceux des systèmes critiques embarqués.

VII – Organisation du mémoire et résumé des contributions

L'organisation du mémoire reprend les étapes de notre démarche en 4 chapitres suivis d'une conclusion et de 7 annexes.

Le chapitre II suivant porte sur l'étude de terrain amont effectuée auprès de 17 familles de la région grenobloise. Au-delà des résultats attendus (vérification de nos hypothèses de travail), cette étude a permis de mettre au point DisQo (Dispositifs du Quotidien), une nouvelle méthode d'investigation qui sollicite l'imagination des participants tout en assurant un juste équilibre entre contrôle expérimental, respect de la sphère privée et validité écologique des résultats (Coutaz et al. 2010).

Le chapitre III porte sur l'état de l'art en matière de PUF et de DUF. Après avoir défini la couverture de ces termes, nous proposons un espace de classification pour une lecture comparative systématique et synthétique des solutions les plus représentatives et notamment des outils portant sur la programmation de l'habitat intelligent. Nous identifions ainsi de nombreuses lacunes avec très peu d'avancées en matière d'aide à la mise au point et à la programmation multisyntaxe. Ces deux aspects conjugués à l'expression des besoins de notre étude amont du chapitre II, serviront d'éléments directeurs à la conception de KISS.

Le chapitre IV présente KISS (Knit your Ideas Into Smart Spaces), un outil DUF, et son infrastructure d'exécution, CONTINUUM (CONTinuité de service en Informatique UbiqUitaire et Mobile). Nous identifions, dans cette infrastructure, les points de contrôle ouverts sur l'utilisateur de façon à assurer un bon équilibre entre l'autonomie du système et le contrôle humain. KISS est l'un de ces points d'intervention. Sa couverture fonctionnelle inclut la programmation, la mise au point et, dans une moindre mesure, la réutilisation des programmes produits par l'utilisateur final. Le langage de programmation est de type déclaratif orienté règles, avec potentiel d'égale opportunité syntaxique entre langue française pseudonaturelle (LPN) et langage visuel iconique. La technique d'interaction de construction des programmes LPN s'appuie sur l'utilisation de menus dont les options sont calculées dynamiquement à partir de l'état de l'habitat intelligent que CONTINUUM actualise en permanence. KISS assure ainsi la découverte progressive du langage ainsi que l'extensibilité et la correction syntaxique et sémantique. Les programmes donnent lieu à une génération de plans d'adaptation dont l'exécution est à la charge de CONTINUUM. La mise au point peut se pratiquer, au choix, dans le monde physique ou dans un monde dual numérique.

Le chapitre V décrit l'expérimentation réalisée pour valider notre approche et évaluer l'acceptabilité de KISS (Fontaine et al. 2012). Afin de concilier pragmatisme et réalisme expérimental, nous avons proposé à treize participants représentatifs de programmer dans DOMUS, un habitat intelligent, une séquence d'activités routinières : le lever matinal, identifié lors de notre étude de terrain amont comme l'un des moments clefs de la vie quotidienne. L'analyse des résultats montre que les utilisateurs adhèrent aux principes de KISS, apprécient l'utilisation du langage pseudonaturel, de même l'utilisation de la réplique virtuelle de l'habitat. Les difficultés rencontrées portent pour l'essentiel sur un vocabulaire jugé trop limité. Un mécanisme de co-construction système-utilisateur est clairement souhaitable, de même le multisyntaxe accompagné de programmation *sur* et *avec* exemples (Girard 2000).

En complément, on trouvera en :

- **Annexe II-1 :** Extrait du rapport du CREDOC

- **Annexe II-2 :** Grille d'entretien auprès des industriels
- **Annexe II-3 :** Scénario directeur pour le développement de KISS
- **Annexe IV-1 :** Grammaire de KISS
- **Annexe V-1 :** Déroulement de l'expérimentation
- **Annexe V-2 :** Questionnaire d'évaluation
- **Annexe V-3** : KISS pour les participants et le magicien d'Oz

Chapitre II : Etude de terrain

Table des matières :

Avant-propos 22

I – Méthodes d'analyse des besoins pour l'habitat 23
I-1 Mesures isolées 23
I-2 Mesures combinées 24

II – Le protocole expérimental 25
II-1 Situer l'étude par rapport au contexte sociétal 25
II-2 DisQo : méthode d'analyse de la perception et des attentes des ménages vis-à-vis de l'habitat intelligent 28
II-3 Recrutement des participants 30

III – Résultats de l'étude DisQo 31
III-1 Confirmation de résultats connus 31
III-1-1 Moments clefs et temporalité 32
III-1-2 Classes de services pour une meilleure qualité de vie 33
III-1-3 Réticences 36
III-2 Résultats originaux 37
III-2-1 Emergence de sens et capacités des objets assemblés 37
III-2-2 Nouvelles fonctions attendues des assemblages d'objets et algèbre d'assemblage 39

IV – Conclusion 41

Résumé :

Ce chapitre décrit l'étude de terrain que nous avons menée pour vérifier les hypothèses, point de départ de notre recherche doctorale. Au-delà des résultats attendus (vérification de nos hypothèses de travail), cette étude a permis de mettre au point une nouvelle méthode d'investigation des besoins et perspectives d'usage pour l'habitat intelligent : DisQo (Dispositifs du Quotidien). Par l'association d'entretiens semi-directifs *in situ* et de sonde culturelle ludique fondée sur l'utilisation d'objets personnels, DisQo sollicite l'imagination des participants (qui doivent se projeter dans l'avenir) tout en assurant un juste équilibre entre contrôle expérimental, respect de la sphère privée et validité écologique des résultats.

Avant-propos

Notre recherche doctorale, rappelons-le, exige de répondre préalablement à deux questions de fond : l'habitant est-il enclin à créer ses propres espaces intelligents ? Si tel est le cas, procédera-t-il de manière incrémentale par assemblage de dispositifs et de services ? Une réponse bien fondée à ces interrogations appelle une étude de terrain au plus près du quotidien des familles. Toutefois, la nouveauté de nos questions exige de définir une méthode qui sollicite l'imagination (puisque les participants doivent se projeter dans des usages futurs) tout en assurant un juste équilibre entre contrôle expérimental, respect de la sphère privée et validité écologique des résultats.

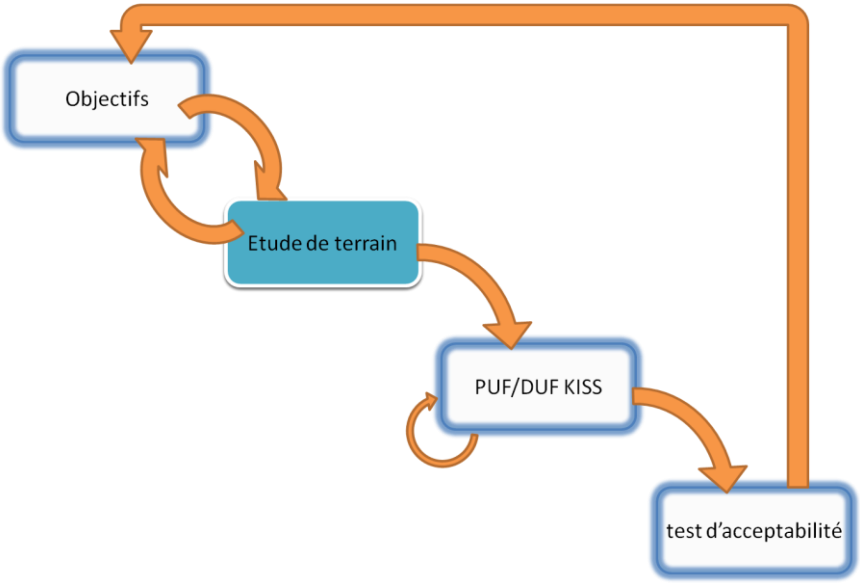


Figure II-1. Notre étude de terrain au sein de notre approche.

Ce chapitre décrit les deux contributions clefs de notre étude de terrain : (1) DisQo (Dispositifs du Quotidien), une nouvelle méthode d'investigation des besoins et des perspectives d'usage pour l'habitat intelligent ; (2) des résultats quantitatifs sur la nature des attentes. Ces résultats confirment des besoins déjà relevés dans la littérature, mais incluent des éléments nouveaux sur le sens que des assemblages de services et de dispositifs sont susceptibles de faire émerger.

Comme le montre la figure II-1, ce chapitre correspond à la première étape de l'approche fixée au chapitre 1. Nous commençons par une revue critique des méthodes utilisées pour l'analyse des besoins dans l'habitat intelligent. Nous en tirons les leçons pour proposer DisQo que nous détaillons en section II. L'analyse des résultats fait l'objet de la section III suivie, en conclusion, du résumé des contributions et des résultats essentiels utiles aux étapes suivantes de notre recherche.

I – Méthodes d'analyse des besoins pour l'habitat

Les sciences sociales comptent de nombreuses méthodes d'analyse des besoins. Dans le contexte de notre étude, nous retenons les méthodes qui impliquent une forte proximité avec le terrain, et notamment avec l'habitat domestique. Par essence, l'habitat domestique est un environnement d'étude non contrôlé et peu contrôlable puisqu'il s'agit de respecter la sphère privée et de veiller à ne pas imposer trop de contraintes tant cognitives que temporelles aux participants. Dès lors, nous restreignons cette brève revue de l'état de l'art aux méthodes qui répondent à ces requis et qui ont été appliquées avec succès en intelligence ambiante, à l'habitat en particulier. Nous les organisons en deux classes : celles qui procèdent par mesures isolées et celles qui utilisent plusieurs sources de mesures.

I-1 Mesures isolées

Une mesure isolée consiste en l'utilisation d'une seule méthode pour aboutir aux objectifs recherchés. La méthode la plus courante est celle de l'entretien avec les utilisateurs cibles. Par exemple, pour concevoir iCAP, les auteurs se sont appuyés sur des entretiens d'une durée de 3 heures chacun en moyenne (Dey et al. 2006). Les entretiens sont semi-structurés de façon à cadrer la conversation sur les objectifs fixés. Hindus complète les entretiens par une phase d'observation prolongée dans l'habitat (Hindus et al. 2001). Cependant, l'absence de support explicite limite l'imagination des sujets. De plus, la présence d'observateurs peut créer un biais, de conformité sociale notamment. Les observateurs font confiance aux membres du foyer qui, volontairement ou non, ne fournissent pas nécessairement toutes les informations pertinentes.

Une solution au problème de la présence de l'observateur est d'introduire une distance entre les participants et lui. Le journal de bord accompagné de consignes de rédaction est une option courante. Par exemple, pour une étude sur l'intégration et l'utilisation des technologies dans les foyers, la consigne consiste à noter toutes les actions relatives à un dispositif représentatif (le magnétoscope par exemple), de noter qui les réalise et dans quel but (Rode et al. 2005). Le journal est restitué quelques semaines plus tard pour analyse. Cette technique n'est pas sans inconvénients : mauvaise compréhension des consignes par les sujets, oubli de rédiger le journal, absence de contrôle des sujets, et durée importante de l'étude.

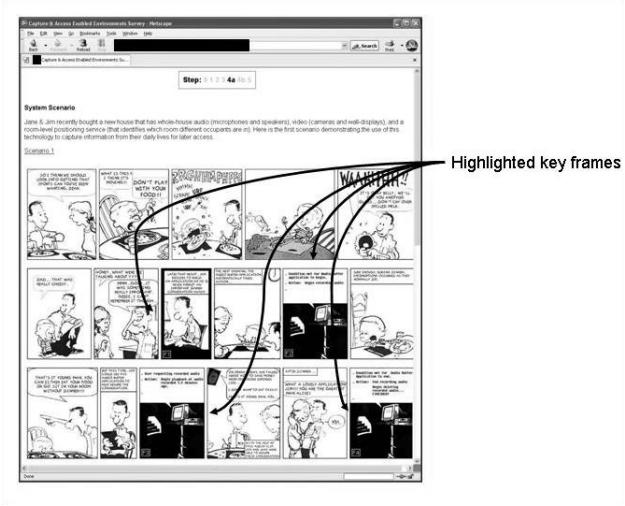


Figure II-2. Matériel d'investigation utilisé pour Camp. Les participants se rendent sur une page internet où leur est présenté un scénario sous forme de bande dessinée. Les participants doivent imaginer ce qui se passe dans les cases surlignées (dites « key frames »).

Dans ces conditions, comment inciter les sujets à effectuer chaque étape consciencieusement ? Bernhaupt propose d'introduire une dimension ludique dans le matériel expérimental (Bernhaupt et al. 2007). Le jeu sert alors de sonde culturelle (Gaver et al. 1999). Il a l'avantage d'impliquer tous les membres du foyer. Toutefois, comme le relève Bernhaupt dans ses expérimentations, certaines familles ne jouent pas ou ne trouvent pas le temps de jouer.

La méthode utilisée pour la conception de Camp conserve la dimension ludique de la sonde culturelle sans imposer de contrainte temporelle (Truong et al. 2004). Le matériel d'étude consiste en une bande dessinée (BD) à compléter sur une page Web en libre accès (cf. Figure II-2). Les visiteurs doivent imaginer ce qui se passe dans les zones vides de la BD. La page Web introduit bien une distance entre les observateurs et les sujets qui n'ont pas conscience d'être observés ou influencés. Cependant, cette méthode ne permet pas d'enregistrer les réactions spontanées. De plus, l'utilisation d'une BD à compléter limite en partie l'imagination fondée sur le quotidien des personnes : les vignettes, qui sont génériques, ne correspondent pas nécessairement au vécu des sujets.

Cette brève analyse des pratiques actuelles représentatives de notre problème indique qu'une seule mesure ne permet pas de répondre à l'ensemble de nos requis. Les mesures combinées apportent des éléments de réponse plus élaborés.

I-2 Mesures combinées

Rode propose une méthode à trois étapes (Rode et al. 2004) : la première consiste à apporter un repas au domicile des personnes participantes et à le partager avec elles. Un lien social est ainsi créé. Dès lors, l'observateur devient l'invité et, dans une ambiance conviviale, les échanges gagnent en naturel et en spontanéité. La deuxième étape consiste à visiter le domicile accompagné des participants en portant l'attention

sur les équipements technologiques et électroménagers et en posant des questions sur leurs utilisations. La méthode se termine par une activité collaborative de placement d'équipements domestiques, introduisant ainsi une dimension ludique. Les contraintes majeures de cette approche sont le temps (il est nécessaire de bloquer une soirée complète), le recrutement (il est délicat de s'inviter chez des inconnus) et l'intrusion (les observateurs s'imposent dans la sphère privée).

Davidoff utilise également la visite des équipements domestiques en compagnie des occupants (Davidoff et al. 2006). Cependant, il oriente les questions liées à l'utilisation de ces équipements sur les moments clefs de la journée comme le matin et le retour à la maison le soir. Dans un second temps, il invite les participants autour d'un jeu de rôle pour identifier la façon dont les membres de la famille organise la planification des activités (Simsarian 2003). Pour finir, il laisse au foyer une sonde culturelle de façon à recueillir des informations complémentaires. Cette sonde permet de détecter sur une semaine les niveaux de stress, de détente et de plaisir autour des deux moments clefs de la journée, ceci sans le biais de la présence de l'observateur. Cette approche pose une contrainte de temps liée à la sonde culturelle ainsi qu'une contrainte d'intrusion liée à la visite de l'habitat. Mais ces deux mesures combinées trouvent un équilibre satisfaisant entre qualité des résultats et contraintes imposées aux participants.

DisQo s'inspire des travaux de Davidoff tout en évitant, autant que possible, ses inconvénients.

II – Le protocole expérimental

Nous avons opté pour un protocole d'étude en deux phases : la première consiste à situer notre étude dans son contexte sociétal de façon à identifier les ménages représentatifs nécessaires à la seconde phase ; cette dernière consiste à étudier les perceptions et les attentes des ménages. C'est ici que nous approchons le quotidien des foyers et que notre méthode DisQo intervient.

II-1 Situer l'étude par rapport au contexte sociétal

Pour situer notre étude dans le contexte sociétal actuel, nous avons commencé par recenser et analyser les données disponibles du Centre de Recherche pour l'Etude et l'Observation des Conditions de vie (CREDOC). Nous nous sommes intéressés au rapport de Novembre 2008[11], « *La diffusion des technologies de l'information et de la communication dans la société française* » (Bigot et al. 2008) dont on trouvera un extrait en annexe II-1. Nous en retenons le taux d'équipement technologique par ménage et l'utilisation qui en est faite.

Pour exemples :

[11] Notre étude a été réalisée durant l'été 2009

- Plus de deux personnes sur trois sont équipées d'un ordinateur et, neuf fois sur dix, elles disposent d'un accès à Internet (Figure II-3). On notera également que 20% de la population possèdent, à domicile, plusieurs ordinateurs[12].

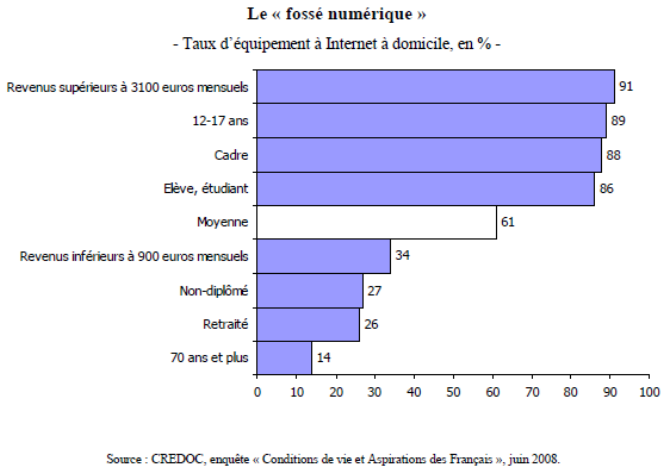


Figure II-3. Rapport du CREDOC 2008, taux d'équipement à internet à domicile en pourcentage.

- Internet occupe une place croissante dans la vie des Français. L'envoi de messages électroniques et la recherche d'informations sont des pratiques courantes. Au cours de l'année 2008, 39% de la population adulte ont effectué des achats par Internet, et 40% ont effectué par ce canal des démarches administratives et fiscales. Le nombre croissant de télédéclarants auprès des Impôts confirme la tendance (7,4 millions de déclarations de revenus par Internet en 2008)[13].

- On observe enfin une progression : (1) de la création de sites personnels ou de blogs (10% des adultes, 53% des adolescents), (2) du suivi de la télévision sur Internet (respectivement 8% et 22%) et (3) de l'utilisation du téléphone via son PC par Internet (9% et 7%).

Ces informations confirment que : (1) les utilisateurs sont de plus en plus équipés en technologies de l'information et de la communication (TIC), (2) la population utilise de plus en plus ces technologies.

Nous avons complété les enseignements encourageants du rapport CREDOC par des entretiens auprès de professionnels dans les domaines des TIC et de l'habitat. L'objectif est d'établir un état des technologies actuelles en intelligence ambiante, de mesurer les attentes du grand public vis-à-vis des TIC, d'avoir un premier aperçu de ce que les foyers possèdent comme équipement TIC ainsi que de connaître les retours utilisateurs sur ces produits. Nous avons organisé les professionnels en deux groupes : les professionnels hautes technologies et les professionnels grand public. De manière à guider ces entretiens, nous avons réalisé une grille d'entretien par groupe de professionnels. Ces grilles sont disponibles en annexe II-2.

[12] En 2011, ils sont trois sur quatre à être équipés d'un ordinateur. 96% d'entre eux disposent d'un accès à internet, et 31% de la population disposent de plusieurs ordinateurs.

[13] En 2011, 48% de la population ont effectué des achats par Internet, 48% y ont effectué leurs démarches administratives.

Hautes technologies

Pour le groupe hautes technologies, nous avons effectué des entretiens auprès de trois sociétés aux métiers bien distincts : Pertech, Aldebaran Robotics et Hilabs.

Pertech est une société spécialisée dans le suivi du regard[14] et dans les tests utilisateurs. L'entretien a débouché sur un scénario d'usage pour un habitat intelligent adapté aux personnes à mobilité réduite. Dans ce scénario, la personne utilise deux modalités pour piloter son habitat : le suivi du regard pour désigner un équipement et la voix pour en activer les commandes.

Aldebaran Robotics vend le robot NAO, un humanoïde programmable de trente centimètres. Cette société a comme projet pour les années à venir de concevoir un robot d'un mètre trente dédié aux services : ouvrir des portes, déplacer des charges, monter des escaliers, assister une personne âgée.

Enfin, Hilabs était[15] une société qui fournissait des vitrines interactives, le passant devenant alors acteur dans le choix du contenu affiché par la vitrine : publicité, informations touristiques, plans de ville, offres promotionnelles, etc.

Bien qu'éloignées de notre cible – l'environnement domestique, ces trois sociétés nous ont fourni un état des technologies relatives à l'intelligence ambiante ainsi que des exemples d'utilisation envisagés dans leur plan de développement.

Grand public

Pour le groupe grand public, nous avons ciblé quatre domaines de professionnels : les vendeurs d'hifi, de téléphonie et d'informatique ; les cuisinistes ; les chauffagistes ; et les régisseurs de télé-alarmes. Les enseignes visitées ont été (dans l'ordre) la FNAC de Grenoble, CORRIOU JCD cuisine, la compagnie de chauffage de Grenoble et leur service de télé-alarme.

En substance, ces entretiens confirment les informations du rapport CREDOC 2008 : le taux d'équipement TIC par foyer augmente très rapidement. Les facteurs importants pour les utilisateurs sont en priorité le gain de temps, l'économie, la simplicité. Les utilisateurs vivent dans l'urgence, font leurs achats sur Internet, etc. Ils veulent que le temps investi sur un système leur apporte du confort ou leur permettent de faire des économies supplémentaires. Ils rejettent l'idée de devoir résoudre de nouveaux problèmes (pannes, dysfonctionnements) ou de devoir faire face à une complexité trop grande pour peu de valeur ajoutée. Le pilotage à distance d'équipements domestiques (chauffage, four, enregistreur vidéo) est considéré comme un moyen de gagner du temps. L'interconnexion d'équipements doit être « *naturelle* » : ajouter une imprimante ou un disque dur ne doit pas être un « *parcours du combattant* ». Des systèmes clef en main, mais personnalisés et adaptés au contexte d'usage, sont un plus. La sécurité des biens et avant tout, des personnes, est au cœur des préoccupations des foyers. Enfin, bien que le coût d'achat soit un argument important, l'esthétisme ainsi que l'effet de mode semblent l'être tout autant.

[14] Eye tracking.

[15] Hilabs a été placée en liquidation judiciaire au cours de l'année 2010.

Situer notre étude dans le contexte socio-économique actuel nous a permis de concevoir la seconde partie du protocole expérimental : l'étude de terrain sur la perception et les attentes des ménages vis-à-vis de l'habitat intelligent.

II-2 DisQo : méthode d'analyse de la perception et des attentes des ménages vis-à-vis de l'habitat intelligent

La méthode DisQo (pour Dispositif du Quotidien) a été conçue pour concilier les besoins de collecte de données écologiquement valides en un minimum de temps tout en respectant la sphère privée des foyers visités. Pour cela, DisQo combine entretiens semi-directifs (Combessie 2007) (Silverman 1997), et sonde culturelle ludique (Bernhaupt et al. 2007). Les entretiens semi-directifs sont nécessaires à la verbalisation des idées ainsi qu'à leur clarification. La sonde culturelle ludique vise à solliciter l'investissement des sujets tout en respectant la sphère privée. Cette combinaison permet de limiter la présence de l'équipe d'expérimentation (de deux ou trois personnes) à 1h30 par foyer.

DisQo comprend quatre étapes successives, dans l'ordre : prise de photos ; mise en situation ; jeu des associations ; débriefing.

Etape 1 : prise de photos
(15minutes)

À l'aide des deux appareils photos numériques fournis par l'équipe d'expérimentation, il est demandé à deux volontaires parmi les membres du foyer de prendre chacun dix photos avec la consigne suivante : « *Il vous est demandé [à chacun des deux volontaires] de prendre deux photos dans cinq pièces distinctes : la cuisine, le salon, la chambre, la salle de bain et une pièce au choix. Dans chaque pièce, il s'agit de prendre en photo un objet utile censé simplifier ou organiser votre vie, et un objet superflu, mais dont il serait difficile de vous séparer. Il vous est également demandé de ne pas parler entre vous ainsi que de ne pas vous retrouver dans la même pièce au même instant*». Ainsi, chaque participant ignore quelles photos ont été prises par l'autre.

Pendant la prise de photos, l'équipe d'expérimentation attend dans le lieu où elle a été accueillie par la famille (généralement au salon), fait quelques échanges informels avec les autres membres de la famille, créant ainsi un lien social. Notons que les expérimentateurs, qui ne visitent pas le domicile, ne s'immiscent pas (ou très peu) dans la sphère privée.

Etape 2 : mise en situation
(20minutes)

Les photos de l'étape 1 servent de matériau[16] à l'entretien semi-directif (enregistré sur support audio et complété par des prises de notes) de la seconde étape. Tous les membres du foyer participent à l'entretien. Pour chaque photo, un expérimentateur demande la raison du choix de l'objet, les valeurs qui lui sont attachées ainsi que les services rendus. Une attention particulière est portée sur l'utilisation des

[16] Les photos sont rapidement transférées sur une tablette par l'un des expérimentateurs.

télécommandes qui prolifèrent et sur les habitudes de programmation des dispositifs du foyer. Les réponses des sujets ont permis de caractériser les objets pris en photos : en particulier, les objets *programmables* (par exemple, machine à laver, réveil, TV), les objets *communicants* (par exemple, téléphone portable, radio, ordinateur), ou bien encore, ceux pour lesquels les sujets ont un attachement *affectif* (par exemple, œuvre d'art, cadeau, meuble). La non-connaissance des photos prises par l'autre participant donne lieu à des échanges riches entre les membres du foyer. Des phrases comme « *Tu as pris ceci, je n'y ai pas pensé »,* ou *« Pour toi, c'est superflu, ça ? »* permettent de recenser les habitudes et pratiques du foyer et de détecter des biais comme la conformité sociale ou la dissonance cognitive. Cette étape de mise en situation conduit naturellement les sujets à avoir en tête des éléments clefs liés à leur mode de vie. C'est sur ces éléments que s'appuie la créativité attendue à l'étape suivante.

Etape 3 : jeu des associations
(45minutes)

Le jeu des associations a pour objectif de stimuler la créativité des sujets en utilisant comme cartes à jouer, les photos des objets personnels de l'étape 1. Un logiciel dédié réalise un tirage aléatoire sur l'ensemble des photos et les présente sur une tablette, deux par deux (couples), puis trois par trois (triplets), avec la consigne suivante : *«Différents objets vont s'afficher à l'écran. Imaginez qu'on puisse les relier pour qu'ils assurent une fonction ou un service qui vous serait très utile ou totalement superflu. Laissez libre cours à votre imagination. Les idées les plus drôles et les plus originales sont les bienvenues.»*

La figure II-4 montre deux exemples de tirage. En raison du caractère aléatoire du tirage, les couples de photos puis les triplets présentés n'ont *a priori* aucun lien entre eux. Ainsi, l'imagination des participants est-elle sollicitée sachant que la créativité augmente avec la distance sémantique entre les éléments d'une même association (Mednick 1962).

Après vingt tirages, un expérimentateur demande aux participants quels autres objets de leur quotidien (pris ou non en photo) ils assembleraient et dans quel but**.** Ces associations libres conduisent à une discussion au cours de laquelle les expérimentateurs recueillent d'autres idées d'associations et d'appréciations sur l'habitat intelligent.

Figure II-4. Deux exemples de tirage aléatoire. « Quel service utile ou superflu apporterait une communication/coopération entre : - votre plante verte et votre grille-pain ? - votre machine à laver et votre téléviseur ? »

Etape 4 : débriefing
(10minutes)

L'étape de débriefing permet de revisiter l'expérience. On s'intéresse ici aux remarques et ressentis sur chaque étape. Ainsi, de nouvelles informations sur les habitudes et les comportements au quotidien émergent. C'est également l'occasion d'obtenir des retours (indirects) sur la méthode DisQo justifiant des points d'amélioration du protocole.

Pour fonctionner, les étapes de DisQo exigent des foyers composés d'au minimum deux personnes. D'autres critères sont également à prendre en compte pour le recrutement des participants que nous présentons ci-dessous.

II-3 Recrutement des participants

Sur la foi des conclusions de Davidoff (*"Families want more control of their lives"*) (Davidoff et al. 2006), nous avons ciblé des foyers qui ont à gérer de nombreuses activités. L'étude préliminaire auprès des professionnels a permis d'affiner la cible en limitant le recrutement à certaines catégories socio-démographiques. Au bilan, les personnes les plus concernées par les dispositifs liés à l'habitat intelligent sont celles dont l'activité professionnelle exige d'organiser et d'ajuster au plus près les activités de chaque membre de la famille. Notre étude s'est par conséquent intéressée aux foyers actifs d'au moins deux personnes dont l'une exerce (ou a exercé) une activité salariée.

Nous avons utilisé DisQo sur un échantillon de 17 foyers, soit 40 sujets au total (35 adultes, 5 enfants). Ces participants ont été recrutés par courriel et par relation personnelle. La méthode des entretiens qualitatifs préconise une vingtaine de

personnes pour arriver à une saturation des propositions et obtenir un éventail d'idées riches et variées (Silverman 1997). Nous considérons donc notre taille d'échantillon (40) comme suffisante.

Parmi les 17 foyers recrutés, 12 sont des foyers à revenu double, 1 foyer monoparental, 2 colocations, 2 foyers de retraités. 15 familles résident en maison individuelle et 2 en appartement. Le tableau II-1 présente la répartition selon les catégories socioprofessionnelles.

Tableau II-1. Répartition socioprofessionnelle des sujets, 35 adultes.

Professions et Catégories Sociales	Nombre de foyers
Agriculteurs	1
Artisans et chefs d'entreprise	6
Cadres supérieurs	6
Professions Intermédiaires	12
Employés-Ouvriers	2
Retraités	4
Etudiants	4

Sur cet échantillon, DisQo a permis de collecter des informations sur plus de 349 assemblages d'objets. Le corpus de données comprend 25 heures d'enregistrement audio accompagnées d'une grille d'analyse pour chaque assemblage et une grille d'analyse thématique sur les pratiques, perceptions et attentes.

III – Résultats de l'étude DisQo

L'étude DisQo confirme les enseignements de l'état de l'art, mais aussi apporte des éléments nouveaux sur la composition d'objets et de services.

III-1 Confirmation de résultats connus

Le corpus DisQo confirme trois classes de faits connus : (1) l'existence de moments clefs dans une journée (Davidoff et al. 2006); (2) des services susceptibles d'améliorer la qualité de vie au quotidien (Truong et al. 2004) et Dey (Dey et al. 2006) ; et (3), en accord avec les données du CREDOC et de notre enquête auprès des professionnels, des réticences et des freins à l'adoption de certaines technologies.

III-1-1 Moments clefs et temporalité

En semaine, les journées des familles actives s'organisent autour de trois moments clefs : le réveil et les préparatifs matinaux, en route sur le chemin du retour au foyer, et l'arrivée au domicile après le travail (Davidoff et al. 2006). Ces moments clefs sont organisés en activités routinières qui ne souffrent pas la perte de temps, ni les exceptions, sources de stress et d'anxiété. Dans les familles dont les deux parents travaillent, l'anxiété monte en puissance lorsque les imprévus surviennent en cascade (Davidoff et al. 2006) au point que les parents ont le sentiment que la vie n'est plus contrôlable (Bandura 1997).

L'expérimentation DisQo confirme les résultats de Davidoff, de même ceux de notre enquête de la phase 1 auprès des professionnels : 30 % des scénarios produits par les foyers ont trait à ces moments clefs, la gestion du temps servant de motivation première. Le concept de *« maison qui se prépare à servir »* suggéré par l'un des participants, reflète et résume parfaitement les attentes.

En plus des moments clefs journaliers, la plupart des scénarios portant sur les assemblages d'objets et de services fait référence à des relations temporelles. Le Tableau II-2 les synthétise et les classe selon les opérateurs temporels d'Allen (Allen 1984).

Tableau II-2. Temporalité dans les assemblages d'objets et de services. Classement et illustration selon les opérateurs temporels d'Allen.

Meet	X Y	*« la machine à café démarre quand je finis ma douche »*
Start with	X Y	*« la musique commence à me suivre dès que j'allume la lumière »*
During	X Y	*« Tout en écoutant de la musique, je me fais aider par mon armoire pour choisir mes vêtements de la journée »*
Before	X Y	*« Je prends une douche puis plus tard dans la journée, mon fauteuil me massera »*
End with	X Y	*« Au moment de partir de la maison, la lumière et la musique s'arrêtent »*
Overlaps	X Y	*« Le chauffage démarre avant la musique du matin et s'éteint durant l'exécution de cette dernière »*
Equals	X Y	*« Le programme télévisé d'une durée identique à celle d'un programme de machine à laver, comme ça quand le programme de la télévision sera terminé, je pourrai étendre mon linge »*

Tous les foyers visités ont produit des scénarios dans lesquels interviennent les relations *meet*, *start with* et *during*. Par contre, *before*, *end with*, *overlaps* et *equals* apparaissent plus rarement. De ce constat, il convient de retenir pour la conception des outils PUF-DUF et notamment pour notre outil KISS (cf. chapitre IV) de favoriser *meet*, *start with* et *during* sans pour autant exclure les quatre autres.

III-1-2 Classes de services pour une meilleure qualité de vie

Comme le montre la Figure II-5, les services résultant d'assemblage d'objets du quotidien imaginés par nos 17 foyers, se répartissent en huit catégories. Ces catégories ont été identifiées à partir d'une analyse thématique (Paillé et al. 2011). Par ordre croissant de fréquence[17] : confort matériel (62,6%), monitoring (27,5%), aide-mémoire/rappel (18,9%), aide à la décision (12,3%), gestion du temps (11%), gestion de stock (9,9%), confort et économie d'énergie (7%), sécurité des biens et des personnes (5,3%). Les services qui relèvent du confort matériel, du monitoring ou des aide-mémoire ont été largement constatés dans la littérature, par Truong et Dey notamment. Dans ce qui suit, nous illustrons ces huit classes de service, puis nous proposons une étude de leurs relations.

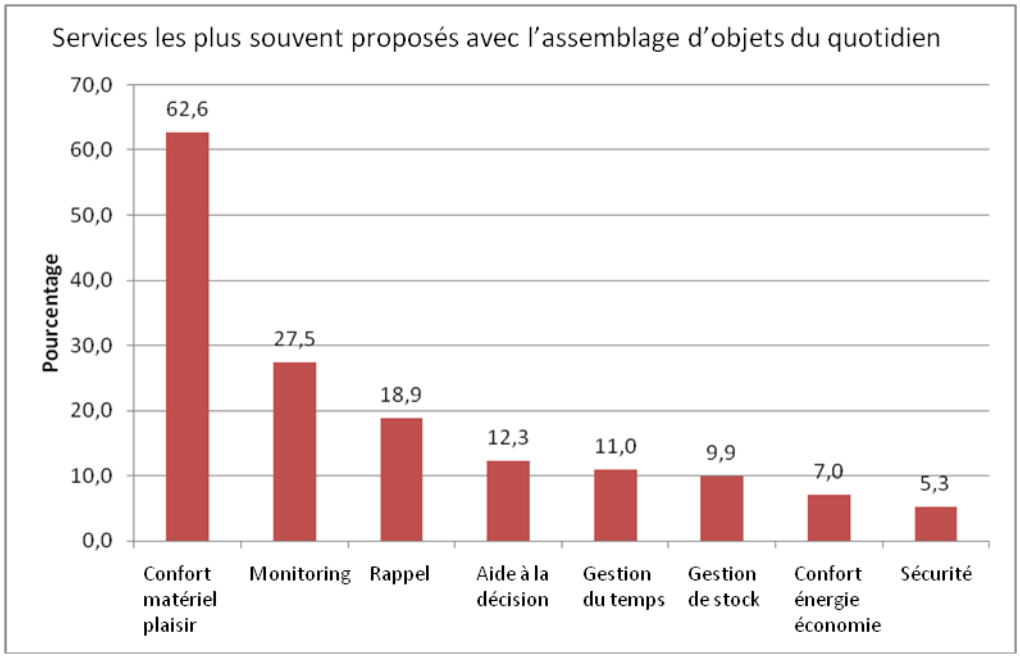


Figure II-5. Répartition des services imaginés par les foyers de l'étude DisQo résultant d'assemblages d'objets du quotidien.

Huit classes de services

- Les services liés au **confort matériel** portent sur l'amélioration du confort et du bien-être par le matériel. Les verbatim suivants illustrent : - le confort pour soi, *« je voudrais de la lumière tamisée »* ; - le confort pour des personnes amies extérieures au foyer, « *j'ai une machine à café, mais je n'en bois pas. Par contre, il m'importe de bien recevoir et mes amis aiment le café* » ; - des demandes floues, « *je voudrais me sentir comme dans un petit cocon* ».

- Les foyers souhaitent inspecter leur habitat (contenu des placards, …) et l'état des équipements (machine à laver, four, …), d'où l'émergence de services de **monitoring**. Ils imaginent effectuer ces observations, de l'intérieur de l'habitat comme de l'extérieur, en situation de mobilité, au travail, ou depuis la douche : « *surveiller le*

[17] La fréquence cumulée est de 154,5%, car plusieurs services peuvent être couverts par un même assemblage.

poulet qui cuit dans le four pendant que je prends ma douche ». D'autre part, la capacité d'observation du système peut être utile pour les aspects de surveillance de l'habitat ou d'économie d'énergie : « *les lumières sont-elles bien éteintes* ? ». « *Ai-je bien fermé la porte à clef* ? ».

- Services de **rappels**[18]. La vie d'une famille est jalonnée de rendez-vous, de « *il faut que je fasse …* », « *… faut pas que j'oublie de …* ». Les foyers souhaiteraient par conséquent se voir proposer par l'intelligence ambiante des alarmes, des rappels, qui les aident à ne pas oublier ces rendez-vous, ces multiples choses à faire et à penser. Le visualisateur d'activités journalières PPTV (Person-Place-Time-View) que propose Davidoff (Davidoff 2012) pour aider les familles à optimiser leur logistique, tend à répondre à ce besoin (cf. Figure II-6).

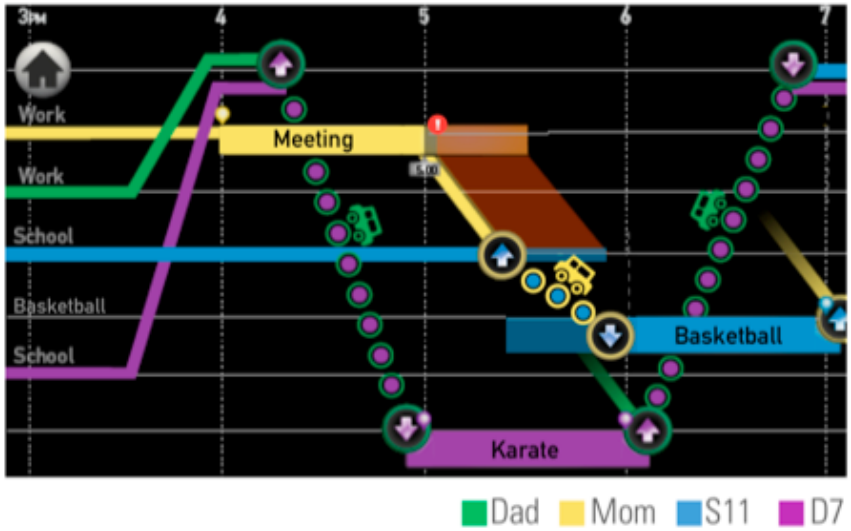


Figure II-6. Le PPTV (Person-Place-Time-View). Chaque ligne représente un lieu clef (école, domicile). Les colonnes dénotent les heures clefs de la journée. Chaque membre du foyer est représenté par une couleur. Une diagonale indique un déplacement entre deux lieux. Les nœuds expriment des choses à faire (prendre ou déposer les enfants à la sortie de l'école) (Davidoff 2012).

- Les services **d'aide à la décision** touchent à l'organisation du quotidien des foyers. En situation de stress ou plus simplement lorsque les foyers n'ont pas d'idée sur que faire ou comment faire, l'intelligence ambiante peut servir de conseiller : « *j'ai des invités qui arrivent dans dix minutes. Parmi ce qu'il me reste à faire, qu'est-ce que je peux faire avant qu'ils arrivent ?* » « *Que puis-je cuisiner en fonction des restes, des aliments bientôt périmés et de l'activité prévue après le repas* ? »

- La **gestion du temps** intervient essentiellement dans les activités des moments clefs de la journée. Mais un autre exemple marquant est la synchronisation des agendas des différents membres d'un foyer : « *Nous menons une vie à cent à l'heure. Entre le sport des petits et nos activités à nous, il nous arrive de nous retrouver tous autour d'une table, les agendas ouverts et de tenter de trouver une date commune pour faire une*

[18] Reminder.

sortie familiale ». On retrouve ici les besoins identifiés par Davidoff et sa réponse technique au problème avec le PPTV.

- Les familles ont à charge la **gestion de différents stocks** : « *linge à laver, à repasser ou à ranger ; nourriture à acheter, à consommer (avant la date de péremption) ; de même pour les médicaments* ». L'un des services attendus de l'habitat intelligent est donc de faciliter la gestion de stocks de nature très différente.

- Confort et économie d'énergie concerne des services dirigés par les ressources communes de l'habitat qu'il convient d'économiser et/ou de partager (l'eau chaude, la salle de bain ou les toilettes). Par exemple, des parents souhaiteraient inciter leurs adolescentes à « *libérer la salle de bain au bout d'un certain temps, par exemple en modifiant le reflet du miroir par une image déplaisante* ».

- La **sécurité** est recherchée pour rassurer sur l'état des occupants et de leurs biens. Il s'agit par exemple de « *prévenir les risques d'intrusion* », de vérifier que personne n'est en danger, que « *les enfants ne se noient pas dans la baignoire* » ou qu'« *il n'y a pas de fuite d'eau* ».

Concomitance de services

Dans l'expression des scénarios, les foyers de l'étude DisQo ont très souvent associé simultanément plusieurs catégories de services. Le Tableau II-3 montre comment. Par exemple, en moyenne les services rendus parlent de sécurité dans 5.3% des cas. Dans 22% des cas, le confort énergétique et économique est associé à la sécurité, et dans 16% des cas au monitoring.

Tableau II-3. Concomitance des classes de services. « ns » signifie « résultat non significatif. Plus la couleur est foncée, plus le service colonne a de chance d'être rendu en complément du service ligne.

Lien entre les services rendus par un couple d'objet (% colonne)	Sécurité	confort materiel plaisir	confort energie economie	rappels	monitoring	gestion du temps	aide à la décision	gestion de stock	Pourcentage moyen
Sécurité		ns	22.0	ns	16.0	ns	ns	ns	**5.3%**
confort materiel plaisir	ns		ns	76.7	86.4	82.0	89.0	93.0	**62.6%**
confort energie economie	21.8	ns		15.0	16.0	ns	17.0	18.0	**7.0%**
rappels	ns	ns	40.6		41.6	48.0	53.6	40.0	**18.9%**
monitoring	83.3	ns	62.5	60.3		ns	57.0	80.0	**27.5%**
gestion du temps	ns	ns	ns	27.9	20.8		26.8	ns	**11.0%**
aide à la décision	ns	ns	31.2	34.9	25.6	30.0		46.7	**12.0%**
gestion de stock	ns	ns	25.0	20.9	28.8	ns	37.5		**9.9%**
Total									
fréquence des services	24	285	32	86	125	50	56	45	
% des services	**5,3%**	**62,6%**	**7,0%**	**18,9%**	**27,5%**	**11,0%**	**12,3%**	**9,9%**	

Ce tableau montre que la majorité des services rendus (62.6%) parlent de « confort matériel et plaisir » (colonne 2). Dans très peu de cas, ces services sont couplés à un

service de « sécurité » (1ère ligne) ou de « confort énergie et économie » (3ème ligne). Les services les plus souvent concomitants avec le « confort matériel et plaisir » (2ème ligne) sont : les services de gestion de stock (93%) ; les services d'aide à la décision (89%) et les services de monitoring (86.4%). Autrement dit, un service « confort matériel et plaisir » satisfait en même temps plusieurs autres catégories de services.

Les services « confort énergie et économie » (3ème ligne) rendent également : un service de sécurité dans 21.8% des cas, un service de gestion de stock dans 18% des cas, un service d'aide à la décision dans 17% des cas, un service de monitoring dans 16% des cas, et enfin dans 15% des cas, un service de rappel d'événements.

Un service de «rappel » (4ème ligne) rend en plus, un service d'aide à la décision (53.6% des cas), et dans 48% des cas, il aide l'utilisateur à gérer son temps.

Les services de « monitoring » (5ème ligne) sont présents à hauteur de 27.5% des cas. Ils rendent aussi, dans 83,3% des cas des services de sécurité, et dans 80% des cas des services de gestion de stock.

La suite du tableau se lit de la même façon. Les services de gestion du temps sont également dans 27.9% des cas des services de rappel, et dans 26.8% des cas, ils aident l'utilisateur dans ses prises de décision. Les services d'aide à la décision sont concomitants à hauteur de 46.7% avec des services relatifs à la gestion de stock, à hauteur de 34.9% avec des services de rappel, et à hauteur de 31.2% avec les services qui touchent au confort énergie et économie. Enfin, les services de gestion de stock servent dans 37.5% des cas, de soutien à la prise de décision.

La connaissance de ces services concomitants peut nous donner un moyen de guider l'utilisateur, ou de lui fournir des conseils sur la suite à donner lors de sa programmation. Cette piste mériterait d'être approfondie.

Si, du point de vue des utilisateurs, l'habitat intelligent et la technologie en général présentent des avancées, ils ne sont pas dénués de connotation péjorative.

III-1-3 Réticences

Les nouvelles technologies ne véhiculent pas toujours une image positive : ergonomie et notices inadaptées, incapacité à comprendre pourquoi « *ça ne marche pas*», etc. Ces classiques ont été largement mentionnés par nos participants. Ont été évoqués la crainte de perdre son autonomie et son indépendance vis-à-vis de la machine (« *je préfère faire l'erreur d'un oubli et apprendre que d'être assisté* ») ou le manque de confiance dans le bon fonctionnement des dispositifs programmables (« *tu programmes, puis quand tu n'es pas là, s'il y a un problème, ta maison est détériorée* »). Néanmoins, nos participants ont estimé que les systèmes fonctionnaient généralement bien, la difficulté majeure étant la prise en compte des situations d'exception : « *La programmation c'est bien dans 90% des cas. Elle va remplir son effet. Mais il suffit d'un moment où je n'ai pas envie de fermer [les volets]. Il faut donc tout ré-ouvrir* ».

L'étude confirme aussi que l'image négative associée aux nouvelles technologies et aux systèmes de programmation qui les accompagnent peut être améliorée si l'utilisateur a un retour sur les gains qu'il peut en tirer, que ce soit en confort, économies d'énergie et plus généralement en terme de gains d'argent et de temps. Si certains avantages, comme le confort, sont perceptibles instantanément, d'autres, comme les avantages financiers, ne le sont qu'*a posteriori*. L'acceptabilité des dispositifs programmables pour l'habitat intelligent passe donc par la présence d'outils de pilotage et de contrôle assurant une connaissance *a priori*, estimée ou réelle, des gains possibles.

III-2 Résultats originaux

Cette section présente les résultats que nous pensons originaux par rapport à l'état de l'art. Ces résultats portent sur la relation entre l'émergence de sens et les capacités des objets assemblés, et sur les grandes fonctions attendues des assemblages.

III-2-1 Emergence de sens et capacités des objets assemblés

Au cours des entretiens semi-directifs, les participants se sont exprimés sur les caractéristiques techniques et sur les valeurs des objets qu'ils ont photographiés. Le jeu des associations, on l'a vu, leur a permis d'imaginer des services futurs. Il paraît donc intéressant d'étudier les relations entre l'émergence de sens, c'est-à-dire la capacité des participants à imaginer des services, et les caractéristiques techniques et valeurs des objets assemblés en termes de capacité de communication et de programmation.

Objets programmables et objets communicants

Dans le contexte de l'intelligence ambiante, deux caractéristiques techniques sont particulièrement pertinentes : l'objet est *programmable* (ou non) et/ou bien l'objet est *communicant* (ou non). Nous obtenons ainsi quatre types d'objets : des objets programmables uniquement – des objets communicants uniquement – des objets à la fois programmables et communicants – des objets ni programmables, ni communicants.

Comme le montre le Tableau II-4[19], l'analyse du corpus DisQo révèle 183 couples d'objets qui ne sont ni programmables ni communicants ; 34 couples dont au moins l'un des objets est communicant ; 108 couples dont au moins l'un des objets est programmable ; et 91 couples dont au moins l'un des objets est à la fois programmable et communicant. Nous constatons que la majorité des objets présents dans l'habitat n'a pas de fonction de communication ou n'a pas de fonction de programmation et que la fonction de communication est la moins présente.

[19] Les 39 fréquences manquantes sont les objets et idées qui ont été proposés de manière spontanée et qui ne rentrent pas dans les traces du jeu des associations.

Tableau II-4. Répartition des assemblages selon leurs caractéristiques techniques (communicants/programmables).

Catégorie du couple	Nombres d'occurrences	Pourcentage
Ni prog. Ni com.	183	43,99
Communicant	34	8,17
Programmable	108	25,96
Les deux	91	21,88

Le tableau II-5 montre le croisement entre la catégorie du couple et l'existence ou non d'un service rendu. Ce croisement permet d'analyser les liens de causalité entre les caractéristiques techniques des objets assemblés et la possibilité que cet assemblage rende un service, c'est-à-dire qu'il y a émergence de sens. Par le test d'indépendance du Chi2, nous avons vérifié que les caractéristiques des objets et l'émergence de sens sont liées de manière significative (chi2= 28,4 p=1%, ddl=3).

Le tableau II-5 se lit ainsi : sur un total de 416 assemblages présentés[20] aux participants, 304 ont fait sens contre 112, soit respectivement 73.1% des cas ont conduit à l'émergence de sens contre 26.9%. La colonne « Communicant » montre que sur un total de 34 assemblages comportant un objet communicant, 28 ont donné lieu à l'identification d'un service contre 6 sans effet, soit 82.4% des cas favorables contre 17.6%. La colonne « Programmable » montre que sur un total de 108 objets programmables, 85 ont donné lieu à l'identification d'un service contre 23 sans effet, soit respectivement 78.7% de cas positifs contre 21.3%. En comparant les valeurs de la ligne « oui », on constate que la capacité d'être communicant influe plus fortement sur l'émergence de sens que la capacité d'être programmable (82.4% contre 78.7%).

Tableau II-5. Croisement entre les caractéristiques techniques des objets et l'existence ou non de service rendu (émergence de sens).

		Caractéristiques du couple d'objets					
Existence d'un service rendu		Ni prog. Ni com.	Communicant	Programmable	Les deux	Total	%
oui	Occurrence	114	28	85	77	304	73,08
	% colonne	62,3	82,35	78,7	84,62	73,08	
non	Occurrence	69	6	23	14	112	26,92
	% colonne	37,7	17,65	21,3	15,38	26,92	
Total	Occurrence	183	34	108	91	416	100
	% colonne	43,99	8,17	25,96	21,88	100	

On note également que 114 assemblages font sens alors que les objets impliqués ne sont ni communicants ni programmables. La valeur que les participants accordent à ces objets est une explication possible.

[20] Le chiffre est gonflé par la catégorisation : un objet de catégorie « Les deux » se retrouve également dans les catégories « Communicant » et « Programmable »

Valeurs des objets

Dans le contexte de ce travail, la valeur d'un objet désigne une sémantique personnelle qu'un participant attache à l'objet. Par exemple : le lit désigne souvent une période (le soir, le matin) ou une situation (le sommeil) ; le lavabo a été photographié plusieurs fois pour évoquer l'eau. De même, les lampes ont été choisies pour évoquer la lumière, les plantes vertes pour évoquer des espaces ouverts, voire le bien-être.

Par exemple, le couple « lit-douche » ne fait intervenir aucun objet communicant et aucun objet programmable. Un nouveau service à valeur ajoutée est néanmoins imaginé et cette valeur n'a de sens que pour un moment particulier de la journée, la matin : « *Quand je me lève du lit le matin, je voudrais que l'eau de la douche coule. Comme ça, quand je sortirai des toilettes, l'eau sera à la bonne température* ».

L'exemple précédent met en exergue deux axes d'analyse : (1) Le moment du réveil fait référence au caractère temporel d'un assemblage. Cet axe sera détaillé dans la section suivante III-2-2. (2) Se lever du lit le matin enclenche le démarrage de la douche : on assiste ici à un enchaînement de services. L'enchaînement de services est l'une des quatre fonctions d'assemblage que DisQo a permise de révéler.

III-2-2 Nouvelles fonctions attendues des assemblages d'objets et algèbre d'assemblage

Au-delà des huit classes de services « classiques » identifiés plus haut (confort matériel, monitoring, aide-mémoire, aide à la décision, gestion du temps et des stocks, etc.), les foyers de l'étude DisQo entrevoient quatre fonctions qui leur seraient utiles : - (1) Remplacer (dans un assemblage préexistant) un service par un autre de meilleure qualité, (2) « augmenter » un objet du quotidien au moyen d'un autre, (3) mettre des services de bout en bout avec enchaînement automatisé afin de gagner du temps dans l'accomplissement de tâches routinières, mais (4) disposer d'un déclencheur pour contrôler les automatismes techniques. Nous présentons en détail chacune de ces quatre fonctions. Nous concluons cette section en généralisant ces nouveaux résultats par l'introduction préliminaire d'une algèbre qui permettrait de raisonner de manière prédictive sur la composition d'objets et de services.

Substitution de service

Plusieurs foyers ont constaté que dans le cas des reportages (notamment, des événements sportifs), les commentaires radiophoniques étaient plus intéressants que ceux de la télévision. En conséquence, les foyers souhaiteraient remplacer le service audio, de qualité informationnelle moindre, de la télévision par celui de la radio. Un autre cas de substitution est le remplacement de l'Interface Homme-Machine (IHM) d'un service fournie par un dispositif donné par celle d'un autre. En l'occurrence, plusieurs participants trouveraient très utile de pouvoir spécifier l'heure de l'alarme du radio-réveil physique via l'IHM de l'horloge de leur SmartPhone qu'ils maîtrisent parfaitement.

Augmentation d'objets du quotidien

Certains objets du quotidien comme le lave-linge ou les placards ne permettent pas d'observer facilement leur état. Plusieurs participants ont suggéré de les augmenter en

les associant au téléviseur dont les ressources écran et télécommande permettraient de visualiser et de contrôler à distance l'objet en question (ce qui éviterait aussi de descendre au sous-sol pour inspecter l'état d'avancement de la lessive alors qu'il est l'heure de partir au travail !).

Enchaînement de services

De façon à gagner du temps dans les tâches routinières, certains foyers ont assemblé des services en séquence, l'activation du second étant conditionné par l'état du premier (relation de causalité). L'exemple du couple « lit-douche » mentionné ci-dessus relève de cette catégorie. D'autres exemples d'enchaînements ont été identifiés comme l'assemblage « douche -machine à café » correspondant au besoin suivant : « *quand je sors de la douche, la machine à café se met en marche et prépare mon café. Comme ça, quand je serai habillé, mon café sera à la bonne température*». Parmi nos résultats, nous avons observé des enchaînements comprenant jusqu'à cinq objets.

Déclencheur de services

Si les foyers souhaitent construire des chaînes de services pour gagner du temps et du confort dans la conduite des activités routinières, ils souhaitent néanmoins en contrôler le lancement : « *99% du temps, la routine se passera bien et le 1% restant va transformer notre réveil en cauchemar* ». En réponse, plusieurs foyers ont imaginé un « package de services» déclenché par un bouton de démarrage installé près du lit, par exemple. L'appui sur ce bouton-starter confirmerait l'exécution du « package » « *réveil du matin* » qui comprendrait le préchauffage de l'eau de la douche, l'éclairage, l'ouverture des stores, ou bien encore l'ambiance musicale et lumineuse. Ce bouton starter sera mis en œuvre dans notre solution technique et expérimenté dans l'appartement DOMUS (Chapitre V).

Vers une algèbre d'assemblage ?

La composition d'objets respecte-t-elle les propriétés de commutativité, de transitivité, d'associativité, de réflexivité ? Si tel est le cas, alors nous pourrions raisonner sur les assemblages d'objets de manière prédictive et/ou explicative. Le corpus DisQo laisse entrevoir quelques promesses dans cette direction, sans toutefois répondre de manière tranchée à cette question. Analysons quelques exemples.

Réflexivité. On pose que tout exemplaire d'objet assemblé avec lui-même est nécessairement lui-même.

Associativité. Le chaînage de services sous-tend la propriété d'associativité. Nous l'avons vu, un participant a qualifié l'assemblage "douche-machine à café" de package du matin. Si l'expression *douche – machine* dénote l'assemblage de la douche et de la machine à café, alors (*douche – machine)*, entre parenthèses, dénote le concept de package. Certains participants, on l'a vu, ont évoqué l'ajout du bouton de démarrage *bouton* afin d'obtenir la chaîne contrôlable *bouton – douche – machine*. En l'occurrence, la construction mentale de nos participants (lancer le package avec un bouton) peut se formaliser par une associativité à droite de l'opération d'assemblage : *bouton – (douche – machine)*. Cette représentation mentale correspond bien à l'effet de la chaîne *bouton – douche – machine*. Autrement dit : *bouton – (douche – machine)* = *bouton – douche – machine*.

Distributivité. L'exemple le plus marquant est l'objet télévision (noté *tv*) qui, nous l'avons vu, met à disposition certaines de ses ressources au profit d'autres équipements comme la machine à laver (noté *m*) ou les placards (noté *p*). Ces assemblages se formalisent ainsi : $tv - (m \mid p) = (tv - m) \mid (tv - p)$ où l'opérateur $\mid$ désigne une alternative ou.

Commutativité. La commutativité a été satisfaite sur la majeure partie des assemblages qui ont fait sens.

Ces informations sur les propriétés algébriques des assemblages méritent une enquête plus approfondie. Nous pensons que ces propriétés peuvent fonder les bases d'une algèbre d'assemblage sur les objets du quotidien.

IV – Conclusion

Pour valider nos hypothèses, nous avons créé une méthode d'analyse des besoins pour l'habitat intelligent : DisQo. Notre méthode combine plusieurs outils de mesure pour aboutir à un équilibre satisfaisant entre contrôle expérimental, respect de la sphère privée et validité écologique, le tout en 1h30 par séance expérimentale.

L'élément clef de notre méthode est la possibilité donnée aux observateurs de visiter l'habitat à travers les images prises par les participants, puis d'utiliser ces photos d'objets personnels comme sonde culturelle ludique. La prise de photos par les participants présente plusieurs avantages : - elle établit une relation de confiance entre les familles et les observateurs ; - les familles révèlent leur habitat et leurs habitudes en s'amusant mais tout en préservant leur espace privé ; - les familles s'impliquent dans l'étude et sont intriguées par la suite de l'expérience. L'utilisation des objets personnels lors du jeu des associations a pour principal avantage de limiter la surcharge cognitive des sujets[21].

Toutefois, DisQo ne permet pas de capter les idées émergentes après coup. En effet, quelques jours après l'étude, certains sujets nous ont fait part de nouvelles idées. Si la possibilité leur était donnée de refaire l'étude, ils prendraient d'autres objets en photo et imagineraient d'autres usages. Par conséquent, nous préconisons de compléter DisQo par un journal de bord ou une sonde culturelle sans toutefois en dépendre.

L'application de DisQo sur un échantillon de 17 foyers a produit plus de 349 assemblages d'objets du quotidien. Nous en avons extrait plusieurs résultats à partir desquels nous pouvons affirmer que les foyers sont prêts à réaliser des assemblages de dispositifs et de services pour améliorer leur qualité de vie. Le scénario d'usage présenté en annexe II-3, illustre une partie des résultats obtenus. Ce dernier, de même l'analyse de l'état de l'art présentée au chapitre suivant, vont guider la conception et l'évaluation de notre PUF-DUF, KISS.

[21] Les participants connaissent les objets. Ils n'ont donc pas besoin de se les approprier.

Chapitre III : Utilisateur final, programmation et génie logiciel : espace problème et état de l'art

Table des matières

Résumé 43
Avant-propos 45

I – PUF-DUF-GLUF : définitions et importance 45
I-1 PUF - Programmation par l'Utilisateur Final 45
I-2 Développement par l'Utilisateur Final et Génie Logiciel par l'Utilisateur Final 46
I-3 Motivations pour des outils PUF-DUF-GLUF 48

II – Espace problème PUF-DUF-GLUF 49
II-1 Classification de l'utilisateur final 50
II-2 Caractérisation de l'environnement de développement et d'exécution 52
II-2-1 Couverture fonctionnelle 52
II-2-2 Support au co-développement 53
II-2-3 Couplage entre phase de conception/développement et phase d'exécution 54

II-3 Caractérisation des langages de développement et outils d'édition 55
II-3-1 Paradigmes de programmation 55
II-3-2 Spécificité contre Généralité 56
II-3-3 Extensibilité 56
II-3-4 Expressivité et adaptation 57
II-3-5 Lisibilité et facilité d'écriture 58

III – Exemples représentatifs de l'Etat de l'art 61
III-1 a CAPpella 61
III-2 iCAP 63
III-3 CAMP 66
III-4 AutoHAN 67
III-5 BYOB appliqué aux dispositifs domestiques 70
III-6 TeC 71
III-7 PANTAGRUEL 75
IV – Analyse synthétique de l'état de l'art 78

Résumé

Ce chapitre vise un double objectif : d'une part, définir les notions clefs de notre étude (utilisateur final, programme et programmer, programmation par l'utilisateur final), d'autre part, analyser l'état de l'art en matière d'outils logiciels pour le développement et la programmation d'habitat intelligent par l'utilisateur final. Nous proposons pour cela un espace problème que nous utilisons comme grille de lecture comparative et systématique des solutions les plus représentatives. Les dimensions principales de cet espace permettent de répondre aux trois classes de questions suivantes :

1. *Concernant l'utilisateur final*, quel rôle est-il supposé assurer ? Est-il un consommateur, un délégateur, un constructeur ?
2. *Concernant l'environnement de développement et d'exécution* des programmes, quelle en est la couverture fonctionnelle au regard des activités de développement (spécification, conception, réutilisation, mise au point, maintenance, déploiement, etc.) ? Cet environnement inclut-il un support au développement collectif et citoyen ? Quel couplage entre phase de développement et phase d'exécution ?
3. *Concernant les langages de programmation*, à quel(s) paradigme(s) obéissent-ils ? Sont-ils extensibles et à quel niveau (lexical, syntaxique, sémantique) ? Sont-ils généraux ou bien sont-ils spécifiques à un domaine auquel cas, observe-t-on une correspondance directe entre les concepts métier et les abstractions du langage ? Y a-t-il structuration en niveaux de complexité ? Comment ces langages améliorent-ils la lisibilité et la facilité d'écriture des programmes : par les modalités de représentation et par les techniques d'interaction (par ex., modalités graphique et textuelle ; techniques d'interaction WIMP/post-WIMP) ? Par emploi d'attributs stylistiques ? Par l'auto-correction? Par une approche multisyntaxe auquel cas, observe-t-on une égale opportunité totale ou partielle entre les syntaxes proposées ? Comment exprime-t-on l'adaptation ? Par paramétrage, par programmation pure (*ex nihilo* ou à partir d'exemples), par composition de composants existants ?

Les réponses à ces questions par l'état de l'art en programmation d'habitat intelligent se résument ainsi : toutes les solutions s'adressent à un utilisateur final constructeur avec absence totale de support à l'expression des requis ou au développement collectif (pas de « place de marché aux composants logiciels » !). L'aide à la mise au point est minimaliste voire absente, et les phases de développement ne fonctionnent pas en fusion avec la phase d'exécution. Du côté langage, si l'extensibilité et l'auto-correction sont plutôt bien couvertes, et si les solutions sont spécifiques au domaine cible, peu d'attention a été portée sur la structuration en niveaux de complexité. Les langages visuels sont bien représentés avec un seul exemple d'approche multisyntaxe et d'interaction post-WIMP de type interface tangible. Notons que l'expression de

l'adaptation par composition est proposée par la communauté scientifique des intergiciels et des systèmes répartis, les autres provenant de la communauté IHM.

Avant-propos

L'étude de terrain présentée en détail au chapitre précédent démontre que l'utilisateur final est disposé à construire des assemblages de dispositifs et de services pour faciliter sa vie au quotidien. Autrement dit, l'utilisateur est susceptible de dépasser son statut de consommateur en faisant de son habitat un lieu de vie agencé sur mesure. Cette construction est envisagée par lui-même de manière incrémentale et opportuniste (Rodden et al. 2004) (Edwards et al. 2001), non pas nécessairement une fois pour toutes par des spécialistes de l'intelligence ambiante. Les outils qui permettent de fabriquer soi-même du sur mesure relèvent de la « Programmation par l'Utilisateur Final » et plus généralement, du « Développement par l'Utilisateur Final » (DUF) et du « Génie Logiciel par l'Utilisateur Final » (GLUF).

Ce chapitre a pour objectif d'analyser l'état de l'art en matière de PUF, de DUF et de GLUF et d'en identifier les forces et faiblesses. Nous proposons pour cela un espace de classification pour une lecture comparative systématique et synthétique des solutions les plus représentatives et notamment des outils portant sur la programmation de l'habitat intelligent. Cette analyse justifie les principes retenus pour la conception de notre outil KISS (Knit your Ideas into Smart Spaces), objet du chapitre suivant. Mais avant de présenter notre espace de classification, il convient de cerner la couverture des termes PUF, DUF et GLUF qui traduisent l'évolution de ce domaine de recherche depuis la fin des années soixante-dix.

I – PUF-DUF-GLUF : définitions et importance

PUF, DUF et GLUF ont en commun l'expression « utilisateur final ». Celle-ci vient des économistes qui distinguent l'acheteur d'un produit de l'utilisateur qui s'en sert (au final !). Dans le contexte de notre étude, cette expression permet de différencier, s'il en est besoin, l'informaticien professionnel, développeur de logiciels destinés au « marché », de la personne qui utilise *in fine* ce logiciel. Nous passons maintenant en revue les trois termes en relation avec le concept d'utilisateur final.

I-1 PUF - Programmation par l'Utilisateur Final

> « Programmer » est un processus qui consiste à traduire, sous forme de programme, une représentation mentale d'un plan d'actions et (ou) d'un ensemble d'états attendus d'une machine dotée de capacité de calcul. À son tour, un programme est un ensemble d'instructions ou d'expressions dont le traitement par cette machine est censé produire ces actions et (ou) états attendus.

Dans le cadre de notre étude, la machine dont il est question est un espace intelligent, c'est-à-dire une réalité mixte où le physique et le numérique se confondent en une machine de traitement de l'information bien plus complexe que l'ordinateur usuel.

> La « programmation par l'utilisateur final » correspond à la situation où les rôles de développeur et de consommateur de programmes sont assurés par la même personne. Autrement dit, la motivation du « programmeur-utilisateur final » est de produire un programme censé résoudre de manière automatisée un problème personnel qui serait plus coûteux à résoudre sans l'aide de la machine. Son objectif premier n'est pas d'apprendre à programmer mais de résoudre ce problème au moyen d'une machine de traitement de l'information (Nardi 1993).

Notre définition rejoint le point de vue de Ko et al. qui définissent la programmation par l'utilisateur final par son intention (Ko et al. 2011), non pas par le noviciat ou l'incompétence supposés de cet utilisateur en matière de programmation[22]. Blackwell, avec son modèle de l'investissement attentionnel (en anglais, « attention investment model »), donne corps à cette intention (Blackwell et al. 2002). En effet, ce modèle structure le processus mental qui conduit une personne à s'impliquer dans une activité de programmation (ou à y renoncer) en fonction des coûts cognitifs à engager comparés aux bénéfices attendus et aux risques potentiels d'un tel programme. Dans le but de baisser ces coûts d'entrée, les chercheurs ont conçu des langages et des techniques de programmation dédiés. Autrement dit, programmer n'implique pas sans toutefois l'exclure, l'utilisation d'un langage de programmation pour professionnel.

I-2 Développement par l'Utilisateur Final et Génie Logiciel par l'Utilisateur Final

Puisque programmer est un processus de production d'artefacts, il convient d'y associer les méthodes, les techniques et les outils susceptibles de faciliter toutes les activités nécessaires à la bonne conduite de ce processus : typiquement, création de programme *ex nihilo*, réutilisation, modification, maintenance, test et débogage, déploiement, bref toutes les activités intervenant dans le développement d'un logiciel.

[22] « We then define *end-user programming as programming to achieve the result of a program primarily for personal, rather than public use* » (Ko et al. 2011).

> Alors que l'on convient d'associer le terme PUF à la phase de création pure d'un programme, le « Développement par l'Utilisateur Final » (DUF) désigne « l'ensemble des méthodes, des techniques et des outils qui permettent, à un moment donné, à des utilisateurs de systèmes logiciels agissant en tant que développeurs non professionnels de l'informatique, de créer, de modifier, ou d'étendre un artefact logiciel.[23] » (Lieberman et al. 2006).

En effet, l'utilisateur final rencontre les mêmes problèmes que le développeur professionnel quand bien même, à l'heure actuelle, ses exigences en matière de processus et de qualité sont, au mieux, implicites. Les développements par l'utilisateur final sont opportunistes et prospectifs (Brandt et al. 2008). Le processus et les requis émergent avec l'expérience, au cours du développement. D'ailleurs, telle était la pratique à l'époque de l'informatique naissante. Mais la tendance au partage et à la co-construction de solutions via les réseaux sociaux (Resnick et al. 2009) (Repenning et al. 2011), conduira tôt ou tard à l'expression de requis extra fonctionnels. Si aujourd'hui personne n'est responsable en cas de mauvais fonctionnement (Kierkegaard 2011), la question se posera inévitablement un jour ou l'autre. On trouvera dans (Ko et al. 2011) des exemples d'outils qui, bien que limités, conduisent les utilisateurs finaux à faire de la qualité malgré eux.

> « Le Génie Logiciel par l'Utilisateur Final » vise à offrir les méthodes, les techniques et les outils de développement qui permettent d'assurer la production de solutions satisfaisant un ensemble de propriétés, ces propriétés définissant à leur tour la qualité requise de ces solutions. Autrement dit, GLUF = DUF + qualité[24]. En outre, le développement, qu'il soit de qualité ou pas, est susceptible d'être une entreprise collective. On parle alors de PUF, de DUF ou de GLUF social, voire citoyen.

Au final, la différence entre le développement logiciel par l'utilisateur final et le développement structuré et systématique par l'informaticien professionnel relève davantage d'un continuum que d'une dichotomie stricte. La figure III-1 traduit ce constat dans un espace à deux dimensions : expérience en programmation d'une part, finalité privée ou publique du logiciel d'autre part. Pourquoi s'intéresser à ces outils de développement ?

[23] Traduction de : EUD is « a set of methods, techniques and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact ».

[24] "End-user programming focuses mainly on how to allow end users to create their own programs, and end-user software engineering considers how to support the entire software lifecycle and its attendant issues." (Ko et al. 2011).

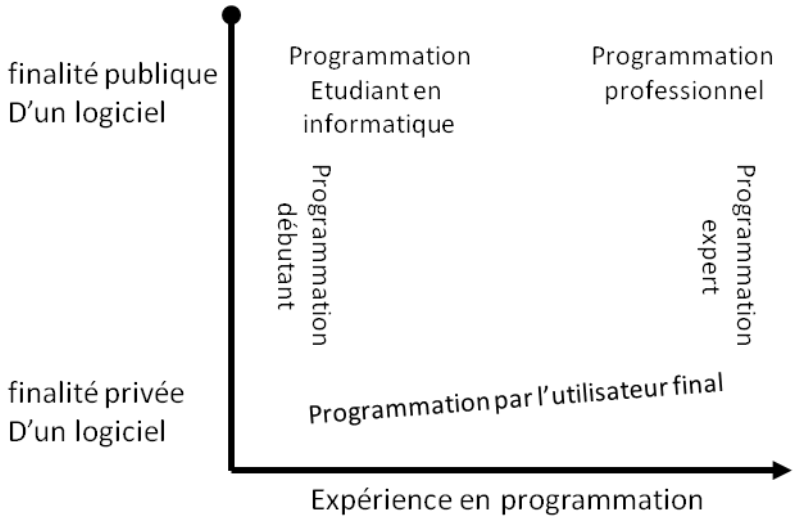


Figure III-1. Les activités de programmation dans l'espace à deux dimensions « expérience en programmation » et « finalité privée ou publique d'un logiciel ». [Traduit de (Ko et al. 2011)]

I-3 Motivations pour des outils PUF-DUF-GLUF

Deux raisons essentielles justifient le développement d'outils PUF-DUF-GLUF : d'une part, les coûts humains et l'évolution permanente des requis de l'utilisateur final, et d'autre part, la pénétration spectaculaire des STIC dans tous les milieux socio-économiques.

Dès 1982, James Martin, dans son ouvrage « Application development without programmers » entrevoit déjà la nécessité de fournir à l'utilisateur final les moyens d'adapter les logiciels qu'il utilise dans ses activités professionnelles (Martin 1982). En effet, il relève que le coût du travail humain est supérieur à celui des ressources de calcul et que, par conséquent, il convient d'améliorer la productivité des employés utilisateurs de logiciels et/ou de se passer progressivement des programmeurs dans les entreprises. Il prône, comme bien d'autres en Interaction Homme-Machine, une approche centrée utilisateur dont on connaît aujourd'hui les limites : les informaticiens ne se donnent pas toujours la peine de maîtriser le métier des utilisateurs cibles et ces utilisateurs ne savent pas toujours exprimer leurs besoins avec précision ni les établir une fois pour toutes. Il en résulte un processus de développement itératif long et coûteux qui aboutit généralement à la livraison d'un produit quelque peu inadapté.

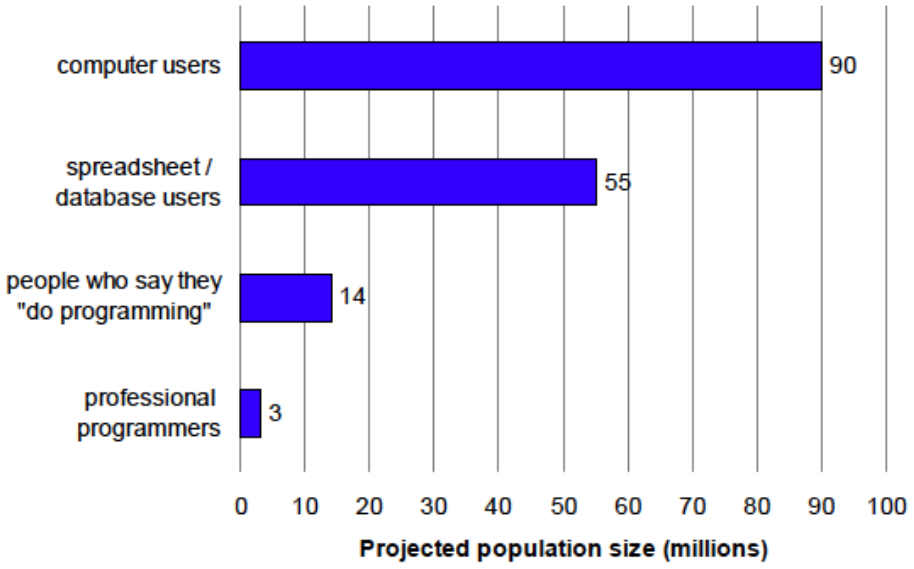


Figure III-2. Projection de la répartition de la population active américaine en 2012. [Tirée de (Scaffidi 2009), p.31, à partir de données publiées par l'État fédéral américain.]

En 2009, Scaffidi complète la tendance que Martin avait relevée près de trente ans plus tôt. Selon lui, en 2012, les utilisateurs d'ordinateurs sur leur lieu de travail seraient 30 fois plus nombreux que les programmeurs professionnels et les personnes qui disent « programmer » seraient 4 fois plus nombreuses que ces mêmes programmeurs (Scaffidi 2009) (cf. Figure III-2). En effet, les utilisateurs de courrier électronique écrivent des règles de façon à filtrer les messages inopportuns. Les physiciens ou les économistes utilisent Matlab pour réaliser des simulations, etc. Or l'analyse de Scaffidi n'inclut pas les activités domestiques. Avec la pénétration massive des STIC dans tous les milieux socio-économiques, l'utilisateur final n'est pas seulement un employé d'entreprise. *Nous sommes tous des « utilisateurs finaux » aux besoins extrêmement volatiles, à l'image de notre société de l'urgence*.

Une réponse, au-delà de l'amélioration des processus de développement, est de donner aux utilisateurs les moyens de faire du « sur mesure » en conformité avec leur investissement attentionnel, sachant que ces utilisateurs sont en nombre considérable, qu'ils sont diversifiés et versatiles. C'est dire l'ampleur du défi ! Nous verrons dans la section III, l'éventail des moyens disponibles que nous analysons selon les dimensions de l'espace problème présenté dans la section qui suit.

II – Espace problème PUF-DUF-GLUF

L'espace problème que nous proposons organise de manière unifiée les aspects clefs de la Programmation par l'Utilisateur Final et de ses extensions, DUF et GLUF, aujourd'hui éparpillés dans la littérature. La figure III-3 montre, sous forme d'arbre, ces éléments de classification que nous reprenons un à un dans cette section. En section

III, Les figures III-24 et III-25 situent dans cet espace des solutions représentatives de l'état de l'art de façon à en faciliter l'analyse.

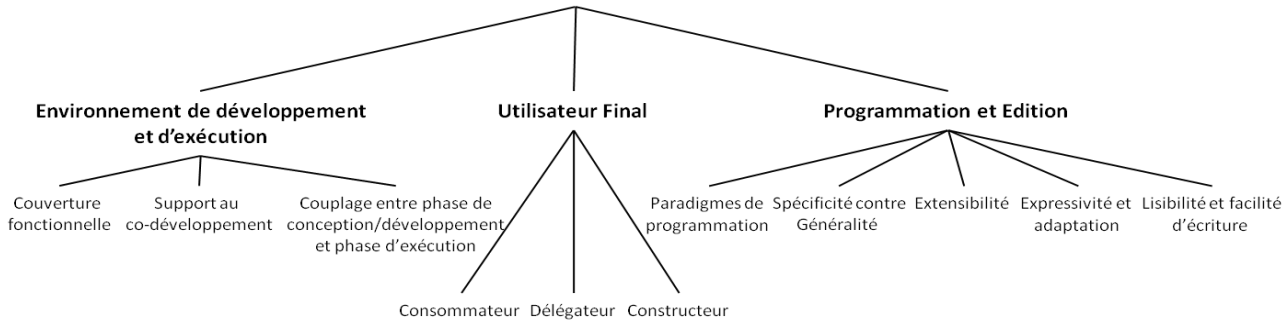


Figure III-3. Les dimensions de l'espace problème PUF-DUF-GLUF selon trois niveaux d'affinement.

Comme le montre la figure III-3, notre espace problème permet de répondre à trois grandes classes de questions :

1. Qui est l'utilisateur final ou mieux, quel(s) rôle(s) est-il susceptible d'assurer ?
2. Comment caractériser l'environnement de développement et d'exécution des programmes de l'utilisateur final ?
3. Quelle est la nature du ou des langages de programmation et des techniques d'édition ? Bien que cette classe de caractéristiques soit un affinement de la précédente, son importance, argumentée ci-dessus, exige de lui accorder une place de premier plan.

II-1 Classification de l'utilisateur final

Dès 1982, Martin caractérise l'utilisateur final en fonction de son degré d'implication dans l'utilisation, directe ou indirecte, d'ordinateurs (cf. figure III-4) et, dans le cas d'une utilisation directe, selon que cette utilisation se fait de manière interactive ou non (appelée à l'époque, « on-line » et « off-line » respectivement). Dans le cas d'une utilisation directe interactive, l'utilisateur final est un pur consommateur de logiciel (il ne crée pas d'application), ou bien il est susceptible de créer des applications selon deux approches : sans programmer c'est-à-dire par spécification, ou bien en programmant avec les langages destinés aux informaticiens professionnels.

Tout récemment, Drey affine cette vision simpliste et dichotomique que Martin fonde implicitement sur le niveau d'expertise en programmation (Drey 2010). Drey propose quatre classes d'utilisateurs finaux en fonction de leur expertise en programmation : l'utilisateur générique qui ne fait qu'interagir avec les applications ; le prescripteur qui a des attentes sans savoir les mettre en œuvre ; l'utilisateur confort qui est capable de paramétrer des applications existantes ; et l'expert métier qui sait concevoir des applications dans un domaine précis avec les outils adaptés.

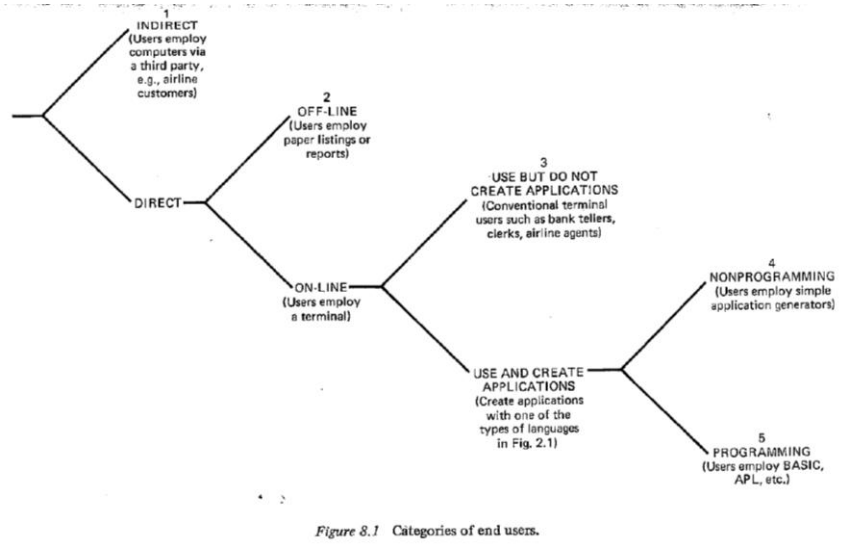


Figure III-4. Classification des utilisateurs finaux selon (Martin 1982).

En cohérence avec le continuum de Ko et al. (cf. figure III.1), nous pensons plus judicieux de classer les utilisateurs, non pas en fonction de leur expertise, mais en fonction de leur rôle. Le rôle d'un utilisateur final est susceptible d'évoluer en fonction des circonstances et en relation directe avec leur investissement attentionnel (au sens de Blackwell, cf. section I ci-dessus), investissement qui dépasse la considération simpliste du seul niveau d'expertise en programmation.

Nous considérons donc qu'un environnement PUF-DUF-GLUF doit pouvoir satisfaire les fonctions nécessaires aux rôles suivants :

- Le rôle de *consommateur* qui se limite à l'utilisation du logiciel tel quel,
- Le rôle de *délégateur* qui a les moyens d'exprimer des requis pour que d'autres les traduisent en programmes en faisant appel à un gourou par exemple,
- Le rôle de *constructeur* qui dispose de tous les outils nécessaires au développement de nouvelles fonctions.

Ces rôles traduisent de manière synthétique les résultats de la recherche en PUF sous l'angle de la production collective. En particulier, Gantt et Nardi, qui ont analysé le processus de développement par des utilisateurs de logiciels de CAO (Conception Assistée par Ordinateur), ont introduit le concept de jardinier (*gardener*) : un expert du domaine (non pas en programmation) qui sert de développeur local et de gourou auprès d'autres utilisateurs de l'entreprise susceptibles, avec l'expérience, de devenir à leur tour des jardiniers (Gantt et al. 1992). Letondal fait le même constat dans son étude auprès de bio-informaticiens (Letondal 2006). L'utilisateur final étant caractérisé par les rôles qu'il est susceptible d'assurer, comment caractériser un environnement PUF-DUF-GLUF de manière orthogonale à ces rôles ?

II-2 Caractérisation de l'environnement de développement et d'exécution

> Un environnement de développement est un ensemble d'outils complémentaires censés structurer les processus de conception et de réalisation de logiciels dont l'exécution est assurée par une machine logicielle et matérielle appelée, dans ce contexte, infrastructure d'accueil ou encore environnement d'exécution.

Étant donné :

- La finalité d'un environnement de développement et d'exécution du point de vue fonctionnel,
- La nature prospective et coopérative du processus de développement par l'utilisateur final,

nous considérons trois classes de questions à propos d'un environnement DUF :

- Quelle est sa couverture fonctionnelle au regard des activités de conception et de développement ?
- Quel support au développement collectif ?
- Quelle forme de liaison entre phase de développement et phase d'exécution ?

II-2-1 Couverture fonctionnelle

Comme le montre la Figure III.3, la couverture fonctionnelle attendue des outils PUF-DUF-GLUF est censée faciliter la conduite des activités tirées du Génie Logiciel : expression des requis, conception et spécification, programmation, mise au point, réutilisation, maintenance et gestion de versions. En l'état de la connaissance actuelle des besoins et des pratiques des utilisateurs finaux en matière de développement, les activités typiques du Génie Logiciel n'ont pas toutes la même importance. Gageons qu'à termes, le partage social exigera l'expression explicite de tous ces aspects. Pour l'heure, nous proposons d'affiner la question d'une activité incontournable : la programmation (cf. section II-3) et sa compagne : la mise au point.

La mise au point inclut d'une part, la conduite de tests pour détecter la présence d'erreur, et d'autre part le débogage qui consiste à identifier les causes d'erreur et à les corriger. Sachant que nous sommes concernés par les espaces intelligents où se confondent les mondes physique et numérique, nous convenons d'affiner plus avant la caractérisation des moyens de mise au point : se fait-elle dans le monde numérique uniquement ? Dans le monde physique uniquement ? Ou bien de façon mixte dans les deux mondes à la fois ?

En effet, si l'on comprend que la mise au point d'une formule de tableur ne peut avoir lieu que dans le monde numérique, il est peut-être souhaitable, au nom du principe de

la « correspondance directe», de tester dans le monde réel une règle dont la fonction est d'agir sur le monde réel (par exemple, fermer tous les volets du domicile). *A contrario*, si la fermeture du volet est conditionnée par le fait que personne ne doit pas se trouver au domicile, alors le test ne peut avoir lieu en présence du développeur dans le domicile en question. Dès lors, le test devrait pouvoir être effectué dans un monde dual numérique avec effet immédiat (ou pas – au choix du développeur) dans le monde physique.

II-2-2 Support au co-développement

Si le processus de développement que l'utilisateur final adopte est quelque peu chaotique, de nombreux travaux ont identifié son caractère collectif, manifeste à diverses échelles :

- Partage social à grande échelle qu'illustre le phénomène emblématique de l'open source et des « app stores » et leurs outils comme Google App Inventor (Collaboration Google et MIT 2010)[25], de même les mashups et leurs outils comme les Yahoo! Pipes[26] (Lin et al. 2008), ou encore CoScripter qui génère automatiquement des scripts pour des tâches répétitives au sein d'applications web (Leshed et al. 2008).
- Co-construction finalisée en interne par une communauté restreinte d'employés d'entreprise, relevant donc d'un métier spécifique : typiquement, en CAO (Gantt et al. 1992) et en traitement de données au moyen des tableurs (Burnett 1999).
- Plus intime, la production familiale à petite échelle, encore peu étudiée en tant qu'activité de groupe (Rode et al. 2005).

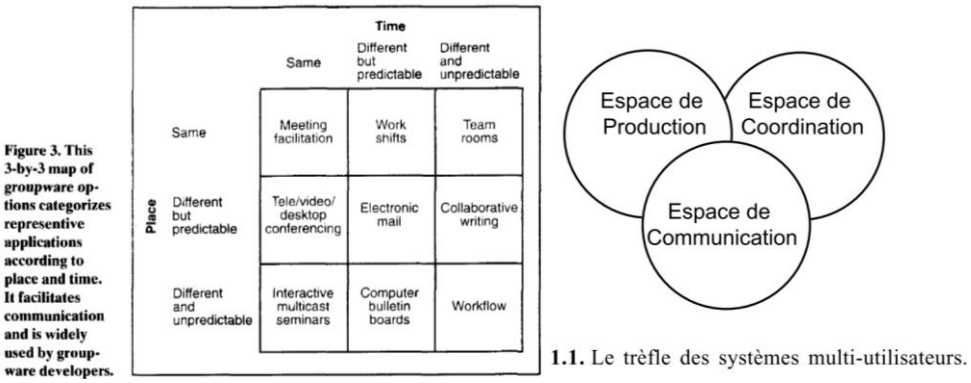


Figure III-5. À gauche, la classification espace-temps de (Ellis et al. 1991) complétée par (Grudin 1994) avec le caractère prévisible ou imprévisible de la localisation dans l'espace et dans le temps. À droite, le trèfle des collecticiels selon la vision fonctionnelle de (Salber 1995).

[25] http://explore.appinventor.mit.edu/
[26] http://pipes.yahoo.com/pipes/

En conséquence, au-delà des rôles que tout utilisateur final est susceptible d'assurer, tout environnement PUF-DUF-GLUF doit pouvoir être caractérisé aussi par ses capacités à faciliter les développements collectifs. Il convient alors d'affiner la dimension multi-utilisateur avec les propositions des deux espaces servant de référence en matière de classification des collecticiels (cf. figure III-5) :

- La classification espace-temps d'Ellis et al. (Ellis et al. 1991) qui permet de distinguer la collaboration en face à face de la collaboration à distance, que celle-ci ait lieu de manière synchrone ou asynchrone, que les lieux et les instants de la collaboration soient prévisibles ou pas (Grudin 1994).
- Le trèfle des collecticiels de Salber et al. qui complémente l'analyse d'Ellis et de Grudin en considérant les trois grandes classes d'activités collaboratives (Salber 1995) : la coproduction de résultats (par exemple, la production d'un programme), la coordination entre les contributeurs pour organiser le travail (ou workflow), et la communication qui permet d'échanger toute information utile à la collaboration que ce soit par l'écrit, l'audio, *via* chat, SMS, etc.

II-2-3 Couplage entre phase de conception/développement et phase d'exécution

Nous avons vu que chez l'utilisateur final, l'expression des requis, les activités de conception, de spécification, de codage et de mise au point, étaient intimement imbriquées. D'où la question de la nature du couplage entre phases de conception et d'exécution. En Génie Logiciel, les environnements de développement usuels présentent une dichotomie claire entre ces deux phases. Avec la nécessité de prendre en compte dynamiquement l'évolution des requis, le Génie Logiciel s'est progressivement orienté vers des processus et des outils plus flexibles qui permettent d'estomper cette distinction. Par exemple, Coutaz et al. utilisent l'ingénierie dirigée par les modèles (IDM) couplée au Service-Oriented Architecture, pour résoudre, à la conception comme à l'exécution, le problème de la plasticité des IHM (Coutaz et al. 2011).

Au vu de l'importance de la co-existence des phases de développement et d'exécution, nous distinguons trois cas :

- Séparation forte entre phases de développement et d'exécution, ce que certains auteurs comme Myers distinguaient par les termes « batch » et « interactif » (Myers 1986);
- Phases confondues : l'utilisateur final ne « voit » pas la différence entre les deux phases en ce sens que l'effet des activités de développement est immédiatement observable dans les résultats de l'exécution. On peut rapprocher ce cas à la « programmation avec exemple » servant par la même occasion d'aide à la mise au point (Girard 2000)[27]. Scratch en est une

[27] Plus précisément, « Un outil de visualisation de programme est dit « avec exemple » si un exemple d'exécution du programme peut être montré à l'utilisateur à l'issue de certaines étapes de construction » (Girard 2000), page 10.

illustration (Maloney et al. 2010). S'il en est besoin, le caractère étroit du couplage entre les phases peut être caractérisé plus avant avec la granularité des unités syntaxiques dont la modification par l'utilisateur a un effet immédiat sur l'exécution.

- Séparation mixte : la séparation ou la fusion est laissée au choix de l'utilisateur final.

Après avoir caractérisé l'utilisateur en termes de rôles, après avoir identifié les dimensions d'un environnement PUF-DUF-GLUF en termes de couverture fonctionnelle, de support à collaboration et de séparation/fusion des phases de développement et d'exécution, nous affinons maintenant le support à l'une des activités clefs : la production de programme proprement dite, ce qui nous conduit à donner une place de premier plan aux langages de développement et aux outils supports.

II-3 Caractérisation des langages de développement et outils d'édition

En informatique, on convient de définir un langage en termes de sémantique, de syntaxe et de vocabulaire. Ces trois aspects traduisent de concert les concepts clefs de base (ou abstractions) que le programmeur doit utiliser (par exemple, le concept d'objet, de règle, de fonction, de processus, ou de service) pour construire une solution qui répond à son problème et qui respecte les règles paradigmatiques du langage.

II-3-1 Paradigmes de programmation

Le(s) paradigme(s) de programmation qu'un langage respecte constitue(nt) un premier niveau de classement. Ainsi, historiquement, distingue-t-on, selon le paradigme de programmation, les langages impératifs et procéduraux (comme Pascal et C), les langages déclaratifs parmi lesquels les langages logiques et les langages fonctionnels, les langages à objets motivés par l'encapsulation et la réutilisation, les langages de programmation orientée aspect motivés par la séparation des préoccupations, les langages multi-paradigmes, etc.

Le Génie Logiciel utilise aussi bon nombre de critères techniques plus fins que le paradigme : par exemple, la nature du typage, la gestion de la mémoire, la réflexivité, le traitement des exceptions et de la concurrence, le type de résolution des liaisons, etc. Si ces critères sont pertinents pour les développeurs professionnels, ils ne nous concernent pas directement (du moins, pour l'instant). Par contre, nous retenons les propriétés susceptibles d'intervenir dans la perception qu'un utilisateur final a du langage : spécificité, expressivité, extensibilité, lisibilité.

II-3-2 Spécificité contre Généralité

L'expérience montre qu'au-delà de la diversité des paradigmes de programmation, on assiste à une explosion de langages spécialisés pour développeurs professionnels : langages de requêtes, langages de description de données, langages formels de vérification, etc. Par définition, les abstractions et les constructeurs d'un langage spécialisé sont conçus pour faciliter l'expression de solutions à des problèmes spécifiques à un domaine ou métier. La programmation par l'utilisateur final n'échappe pas à cette règle. Nous considérons deux caractéristiques complémentaires :

- Correspondance directe (ou expressivité),
- Structuration en niveaux de complexité croissante.

En bon respect des principes fondamentaux de l'Interaction Homme-Machine, un langage PUF doit être caractérisé par la correspondance directe (ou non) entre les abstractions et constructeurs qu'il offre et les concepts et la structure des tâches humaines dans le domaine cible (pour nous, l'habitat). Une correspondance directe signifie que le langage n'introduit pas de bruit qui viendrait détourner l'attention de l'objectif. En ce sens, un langage à correspondance directe est plus expressif qu'un langage pour lequel il n'y aurait pas cette « directness ». Par exemple, les tableurs n'imposent pas de déclarer des variables ni de compiler pour obtenir la somme des valeurs d'une colonne (Lewis et al. 1987). Concernant plus particulièrement les langages de programmation pour l'habitat, notre étude de terrain a mis en évidence le besoin d'exprimer des relations temporelles (cf. section III-1-1 du chapitre II). Il importe donc, au nom de la correspondance directe, qu'un langage de programmation pour l'habitat soit caractérisé par sa capacité à traduire les relations d'Allen.

De manière orthogonale au caractère direct de la correspondance langage-tâche, le langage (ou l'environnement) permet-il une découverte du langage qui soit progressive, sans barrières cognitives infranchissables ? Cette question est en relation avec le modèle de l'investissement attentionnel. Autrement dit, les tâches simples et courantes peuvent-elles être programmées d'emblée sans erreur avec très peu d'efforts cognitifs - *low threshold* (Myers et al. 2000), puis l'expérience aidant, l'utilisateur final peut-il découvrir des primitives plus puissantes qui permettent de répondre à des problèmes plus complexes - *high ceiling* (Myers et al. 2000) sans que le passage d'un niveau à l'autre exige des efforts incompatibles avec l'investissement attentionnel - *vertical walls* (Myers et al. 2000)? On rejoint ici les *training wheels* que Carroll a appliqués comme principe de conception des Interfaces Homme-Machine (Carroll et al. 1984)[28].

II-3-3 Extensibilité

Nous considérons l'extensibilité d'un langage PUF car, bien qu'il soit spécialisé, cette propriété signifie offrir des moyens d'ajustement du langage au plus près, mais aussi

[28] Dans la vie réelle, les vélos des enfants sont équipés de stabilisateurs pour éviter les chutes. Dès que l'enfant maîtrise l'équilibre, les stabilisateurs sont retirés. Par analogie, Carroll propose d'organiser les fonctions d'un système en niveaux de complexité croissante, chaque niveau définissant un garde-fou cognitif.

d'en élargir la généralité et donc le domaine d'exploration – que Resnik et al. désignent par *wide walls* (Resnick et al. 2005). L'extensibilité est d'autant plus intéressante que l'utilisateur final d'un habitat intelligent voit ses besoins évoluer. Par exemple, un nouveau dispositif dont le type est inconnu de l'environnement, doit pouvoir être manipulé par programmation.

On distingue trois niveaux d'extensibilité :

- extensibilité lexicale : est-il possible d'introduire de nouveaux symboles ?
- extensibilité syntaxique : est-il possible d'introduire de nouveaux concepts et/ou de nouveaux constructeurs ?
- extensibilité sémantique : est-il possible d'introduire de nouveaux comportements primitifs ?

II-3-4 Expressivité et adaptation

Dans le contexte de notre recherche, nous l'avons vu au chapitre II, l'utilisateur final a pour objectif central d'adapter l'habitat intelligent à ses besoins, de faire du sur mesure. Pour cette raison, nous considérons la puissance d'expression d'un langage par le type d'adaptation (de personnalisation) qu'il permet d'exprimer.

En Interaction Homme-Machine, l'adaptation est considérée sous deux angles complémentaires : l'adaptabilité et l'adaptativité. L'*adaptabilité* désigne la capacité de personnalisation par l'utilisateur final tandis que dans le cas de l'*adaptativité*, l'initiative ou l'anticipation de l'adaptation revient au système. L'adaptativité s'appuie le plus souvent sur des techniques d'apprentissage par le système. Dans le contexte de notre discussion sur les langages PUF, nous sommes concernés par l'adaptabilité.

Nous proposons quatre niveaux d'adaptabilité, par ordre croissant de souplesse :

- Adaptabilité par *paramétrage* : l'utilisateur final spécifie ses choix parmi les options prévues par les développeurs du système. Il n'y a pas création de la part de l'utilisateur. Les unités d'exécution, ou composants du système, produites avec le langage exportent des paramètres accessibles à l'utilisateur final et seulement cela.
- Adaptabilité par *composition* : le langage permet à l'utilisateur final de créer une solution par composition, ou par reconfiguration, de composants existants. Il n'y a pas création de composants, mais création ou modification de comportement par sélection et assemblage de composants prêts à l'emploi.
- Adaptabilité par *programmation*, soit *ex nihilo*, soit par imitation ou modification d'exemples. Alors, il faut distinguer :
    - Le codage direct explicite à partir de rien ou d'exemples (gabarits) modifiables.
    - Le codage par spécification et génération de code à partir de rien ou par généralisation à partir d'exemples ou, pour reprendre l'expression de P. Girard, « sur exemples » (Girard 2000). À son tour, le code produit peut, être ou ne pas être, accessible à l'utilisateur final, donc

être ou ne pas être modifiable. Un code généré modifiable offre l'avantage d'une personnalisation fine, mais au risque de perdre la conformité avec la spécification source si l'environnement n'offre pas les transformations inverses.

- Adaptabilité par extension : le langage permet d'étendre le comportement par intégration de composants produits par des tiers et/ou par l'utilisateur final.

> Nous avons utilisé le terme de composant au sens large : une unité de calcul réutilisable. En pratique, celle-ci peut s'instancier sous forme de gabarit, de patron, de module, de composant orienté service, de device UPnP, etc.

Les propriétés vues jusqu'ici permettent de caractériser l'adéquation des primitives et de la structure profonde du langage aux besoins centrés tâches de l'utilisateur final. Il nous reste à considérer la surface du langage telle que la perçoit l'utilisateur final sous l'angle de la lisibilité et de la facilité d'écriture.

II-3-5 Lisibilité et facilité d'écriture

La lisibilité et la facilité d'écriture de programme, qui portent sur la surface perçue et manipulable des programmes par un humain, sont directement véhiculées par l'outil d'édition. Cet outil est un cas particulier d'éditeur de documents, un programme étant un document dont la structure est fortement contrainte. Dès lors, nous distinguons quatre axes d'analyse :

- La capacité de l'éditeur à favoriser la *construction de programmes corrects*,
- La capacité de l'éditeur à proposer *plusieurs syntaxes*. Dans ce cas, quelles relations ces syntaxes entretiennent-elles ?
- Pour chaque syntaxe, la (ou les) *modalité(s) de représentation* des abstractions et des constructeurs du langage de programmation,
- La (ou les) *technique(s) d'interaction* ou paradigme(s) pour éditer un programme : manipulation directe, interaction langagière, interaction tangible et ses dérivés comme « graspable user interfaces », Reality-Based user interfaces (Jacob et al. 2008), etc. Nous ne reviendrons pas sur la nature de ces paradigmes largement débattue dans la littérature.

L'*aide à la construction de programmes corrects* en facilite l'écriture puisqu'elle élimine d'emblée certaines erreurs humaines, évitant *de facto* de nombreux tests et corrections préliminaires. Nous reprenons à notre compte la distinction classique de la correction d'un programme aux niveaux lexical, syntaxique et sémantique. Si les supports à la correction lexicale et syntaxique sont devenus courants avec les éditeurs syntaxiques, la correction sémantique est plus difficile à assurer : un programme peut être sémantiquement correct, mais son exécution n'a pas nécessairement les effets attendus dans le contexte d'exécution actuel.

Approche multisyntaxe. Cette approche consiste à intégrer dans un même éditeur de programme plusieurs syntaxes sémantiquement équivalentes. L'utilisateur final a alors

la possibilité de construire un programme avec la syntaxe de son choix. Puisqu'il y a équivalence sémantique, on peut affiner plus avant le multisyntaxe avec l'existence (ou l'absence) partielle ou totale de transformations entre les syntaxes. L'utilisateur peut-il éditer son programme en utilisant la syntaxe de son choix et le système produit-il le programme équivalent dans les autres syntaxes disponibles et vice versa ? Dans ce cas, la *propriété d'égale opportunité entre les syntaxes* est satisfaite. Si les syntaxes sont étanches, l'utilisateur a le choix de la syntaxe qui lui convient, mais il ne peut pas passer de l'une à l'autre de manière quelconque. Par exemple, AutoHan propose trois syntaxes que l'éditeur est capable de traduire dans un même langage pivot dit *Lingua franca*. Nous verrons dans l'analyse de l'état de l'art que la propriété d'égale opportunité n'est que partiellement supportée par AutoHan (Blackwell et al. 2002).

Concernant les *modalités de représentation*, nous partons de la définition du concept de modalité proposé par Nigay comme étant le couple <*langage de représentation, dispositif*> (Nigay 1994) où « langage de représentation » désigne un système conventionnel structuré de signes qui assure une fonction de communication, et où « dispositif » désigne tout périphérique d'une machine pouvant servir de support (en sortie) ou d'actionneur (en entrée).

Prenons trois exemples : les Media Cubes, Scratch, et JigSaw (voir figures III-6). Dans le cas des Media Cubes, une phrase est un assemblage linéaire de cubes physiques construite par manipulation (ultra-directe) des cubes. Chaque face de cube est un dispositif qui sert de support de rendu à un mot terminal du langage, ce mot pouvant être représenté soit sous forme textuelle (par ex. STOP), soit sous forme iconique (par ex. ⏰). On a là un langage dont les terminaux sont textuels ou iconiques et dont le support de rendu est purement physique. Une phrase a pour représentation un assemblage linéaire de cubes physiques placés côte à côte, orientés correctement et couplés à des entités du monde réel pour faire sens. (Nous verrons plus loin, en III-4, la question du couplage).

Avec JigSaw, une phrase est également un assemblage linéaire, mais cette fois d'icônes numériques qui utilisent un même écran comme support de rendu. Chaque icône représente un mot terminal du langage (en l'occurrence des services). Dans une version matérialisée de JigSaw, les icônes numériques sont remplacées par des pièces en carton. Le langage de représentation est le même que dans la version numérique (tous deux obéissent à la même syntaxe). Seule change la nature physique/numérique des dispositifs servant de support au rendu. Nous assistons à deux modalités de représentation.

Dans Scratch, tous les éléments du langage sont de nature iconique ou textuelle. Mais contrairement aux Media Cubes et à JigSaw, l'assemblage de ces éléments exploite les deux dimensions d'un plan pour faire sens. Pour cette raison, on parle de langage visuel. Mais comme pour JigSaw, le support de rendu des éléments du langage et des

programmes est un même écran. Scratch, tout comme JigSaw et les Media Cubes, applique les principes de la manipulation directe, mais via une souris.

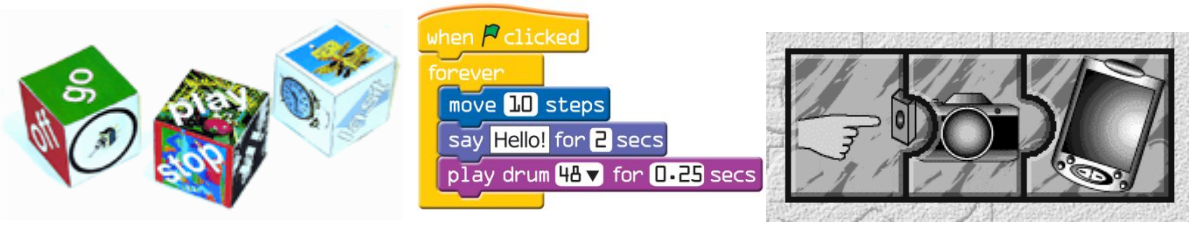


Figure III-6. De gauche à droite, un programme Media Cubes (Blackwell et al. 2002), Scratch (Maloney et al. 2010), JigSaw (Humble et al. 2003).

Notons qu'en Génie Logiciel, il existe deux techniques complémentaires bien connues pour améliorer la lisibilité d'un programme : l'utilisation de sucre syntaxique et la définition d'une stylistique idoine. Ces aspects permettent d'affiner, s'il en est besoin, chaque modalité de représentation.

- *Sucre syntaxique*. En matière de langage de programmation, on appelle sucre syntaxique les extensions apportées à la syntaxe d'un langage pour améliorer la lisibilité et l'écriture des programmes sans en changer la sémantique. Par exemple, la possibilité d'écrire *print a, b, c;* à la place de *print (a, b, c);* simplifie l'écriture de l'instruction *print* par l'élimination des parenthèses.
- *Stylistique*. La stylistique d'un langage relève de la même motivation mais sans modification syntaxique. Elle porte sur les conventions d'écriture. Par exemple, Sale propose, pour le langage Pascal, une stylistique qui recommande de noter les mots clefs du langage en caractères gras et les commentaires en italique (Sale 1979). Les conventions de nommage, l'indentation ou le retour à la ligne (lorsqu'ils ne servent pas de séparateur) sont également des éléments de stylistique.
- Les extensions syntaxiques et les conventions stylistiques sont souvent intimement liées dans les outils d'édition. Par exemple, dans le langage Scratch, conçu pour un utilisateur final supposé naïf, les formes graphiques des instructions et les couleurs sont utilisées comme marqueurs syntaxiques et stylistiques (voir figure III-7). JigSaw encadre les mots terminaux du langage par une courbe spline fermée en forme de pièce de puzzle pour soutenir l'esprit de la métaphore.

Figure III-7. Un exemple de programme en langage Scratch. Chaque instruction « if » usuelle est augmentée de marqueurs graphiques pour indiquer sa nature et sa portée. L'entête de cette instruction dispose d'un emplacement pour accueillir une valeur booléenne représentée graphiquement par un hexagone, son corps accueille d'autres instructions assemblées telles des pièces de puzzle. Ce programme signifie : si le chat est touché, dire « Nice kitty ! » pendant deux secondes, puis jouer le son meow (miaulement).

Ce qui suit est une analyse de l'état de l'art présentée selon les dimensions de notre espace problème PUF-DUF-GLUF.

III – Exemples représentatifs de l'Etat de l'art

Nous sommes concernés par les outils qui donnent à l'individu le pouvoir de résoudre des problèmes qui lui sont propres par le biais de la programmation. En conséquence, ne rentrent pas dans notre épure, les outils de programmation comme Karel (Pattis 1981), Alice (Conway et al. 1997) conçus pour apprendre à programmer[29]. Notre analyse cible en priorité les outils DUF portant sur les espaces intelligents. En gros, nous trouvons deux types de propositions :

- Les outils émanant de la communauté IHM conçus selon une approche centrée utilisateur. Il s'agit de a CAPpella, iCAP, CAMP, AutoHAN et, dans une certaine mesure, BYOB.
- Les outils émanant de la communauté systèmes répartis et intergiciels. Il s'agit de TeC et de PANTAGRUEL.

III-1 a CAPpella

a CAPpella est un environnement de prototypage d'*applications sensibles au contexte* (Dey et al. 2004). Par analogie avec les chants exécutés sans accompagnement instrumental, a CAPpella permet à l'utilisateur final de programmer sans l'accompagnement de professionnels. L'utilisateur visé est donc, dans notre

[29] On trouvera dans (Pausch et al. 2003) une présentation approfondie de l'état de l'art sur les langages conçus spécifiquement pour apprendre à programmer.

classification, un *constructeur*. Ici, programmer consiste à faire générer par le système des reconnaisseurs de situation (par exemple, reconnaître le début d'une réunion) et, pour chaque situation reconnue, effectuer les actions attendues (par exemple, allumer ou éteindre la lumière). Si le domaine d'application est assez large (*applications sensibles au contexte*), aCAPpella n'offre *aucun support au co-développement*.

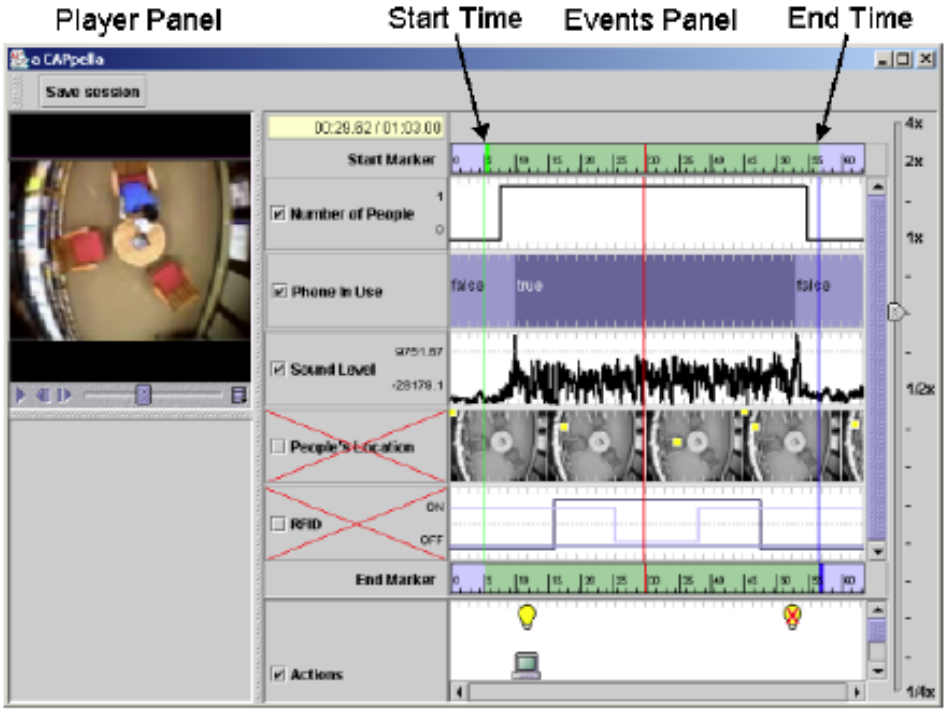


Figure III-8. L'IHM de a CAPpella en mode édition (programmation). À gauche, les données brutes audio et vidéo que l'utilisateur peut rejouer. Le panneau de droite comprend deux parties : d'une part, les sources d'informations (les variables du programme comme le nombre de personnes dans la pièce) et leurs valeurs successives enregistrées pendant la session ; d'autre part, dans la ligne du bas, les actions effectuées pendant cette session (ici, lumière allumée et lancement du programme d'enregistrement de notes en début de session, lumière éteinte en fin de session). Le temps est représenté horizontalement, de gauche à droite. L'utilisateur vient de marquer le début et la fin de la session et souhaite exclure les données en provenance des RFID, de même celles en provenance du service de localisation de personnes.

a CAPpella fonctionne de manière cyclique selon deux modes : (a) en mode enregistrement, il enregistre à la volée les données du monde réel fournies par les capteurs environnementaux (audio, vidéo, RFID, lancement/arrêt de services, etc.) et détecte les événements en faisant appel à ses reconnaisseurs ; (b) en mode édition, l'utilisateur améliore les capacités de reconnaissance du système en introduisant des annotations (par exemple, en marquant le début et la fin d'une réunion). a CAPpella, nouvellement informé améliore ses connaissances. Il est ensuite utilisé à nouveau *in situ* en mode enregistrement, puis en mode édition jusqu'à ce que son comportement donne satisfaction. L'existence de ces deux modes indique une *séparation forte* entre phase de conception et phase d'exécution.

Concernant sa couverture fonctionnelle, a CAPpella s'appuie sur la *réutilisation des services déployés* par l'infrastructure (localisation de personnes, détecteurs

d'événements). La mise au point est mixte, soit dans le *monde numérique* en rejouant la session en mode édition, soit dans le *monde physique in situ*, mais, semble-t-il, pas dans les deux mondes en même temps.

Le langage de programmation obéit au *paradigme déclaratif* de règles (avec en partie gauche, la description d'une situation, et en partie droite, les actions à effectuer). Le langage est *spécifique au domaine des applications sensibles au contexte*. Si a Cappella ne propose *pas de structuration en niveaux de complexité*, Dey et al. ont porté toute leur attention sur une conception qui assure une *correspondance directe* entre les concepts du langage et les besoins des utilisateurs. En particulier, l'expression de relations temporelles est possible, bien que ce support ne soit que très partiel (marquage de début et de fin de session uniquement).

Les services de l'infrastructure d'exécution pouvant évoluer, le *langage est potentiellement extensible, du lexique à la sémantique*. L'utilisateur fabrique du sur mesure *par extension* : il réutilise tout ou partie des services de l'infrastructure qu'il assemble pour spécifier de nouvelles situations ainsi que les actions attendues lorsque ces situations se présentent. Il s'agit d'une programmation, non pas *ex nihilo*, mais *à partir d'exemples* enregistrés *in vivo* par le système. Cette programmation se fait par *spécification* conduisant à une *génération de reconnaisseurs par inférence* (apprentissage).

Les modalités de représentation et d'interaction pour construire un programme est typiquement *WIMP à manipulation directe*, mais a CAPpella n'est *pas multisyntaxe*. Comme le montre la figure III-8, les variables (nombre de personnes, niveau sonore, etc.) ainsi que les actions possibles sont représentées graphiquement. Elles sont automatiquement déclarées ce qui constitue une aide à la production de *programmes syntaxiquement et sémantiquement corrects*.

Dey et al. estiment que a CAPpella relève de la programmation par démonstration. En effet, l'utilisateur enregistre des informations contextuelles *in situ* pour alimenter les mécanismes d'apprentissage. Toutefois, lorsque l'utilisateur annote le programme, il ne procède plus par démonstration.

III-2 iCAP

iCAP (interactive Context Aware Prototyping) partage de nombreux traits avec a CAPpella (Dey et al. 2006) : même domaine d'application (*applications sensibles au contexte* et notamment l'*habitat intelligent*) ; l'utilisateur visé est un *constructeur* ; comme pour a CAPpella, *absence de support au co-développement* ; même paradigme de programmation (*déclaratif par règles*) avec un effort marqué pour une *correspondance directe* et *sans structuration apparente en niveaux de complexité*.

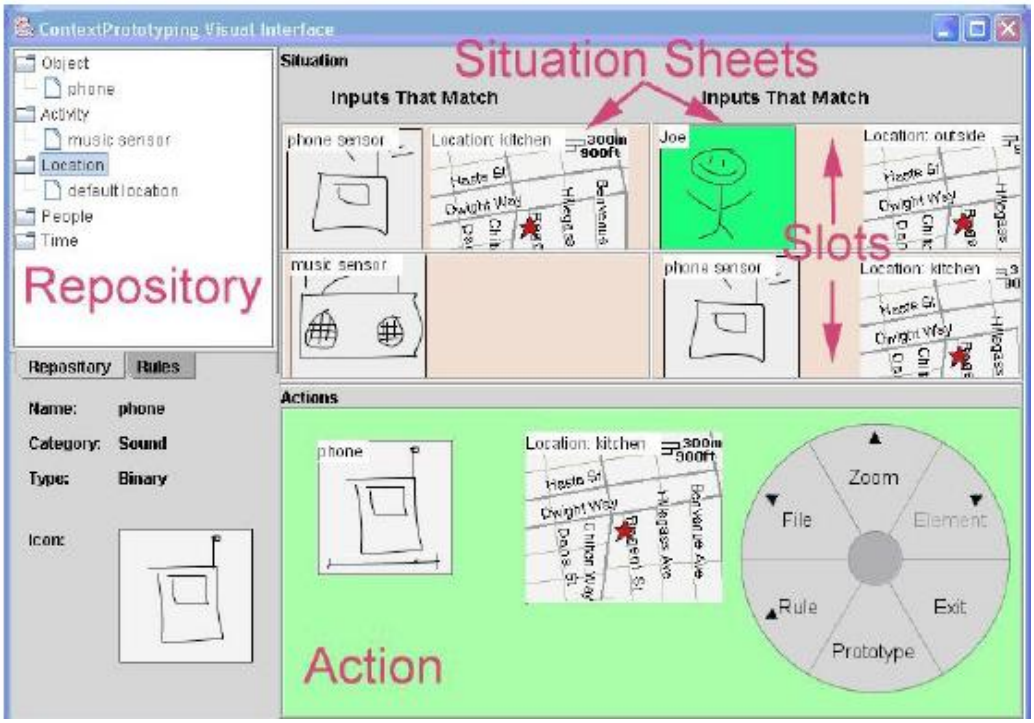


Figure III-9. L'IHM de iCAP. À gauche, la représentation des concepts du langage (le lexique et les constructeurs) à savoir : les objets physiques (par ex., le téléphone), les activités, les lieux (cuisine, etc.), les personnes, l'heure. L'utilisateur vient de s'intéresser à l'objet « phone » dont il a dessiné l'icône de représentation (en bas, à gauche). Le panneau de droite est dédié au programme et comprend deux parties : (a) en haut, sur fond beige, la spécification d'une situation sous forme de règle (en l'occurrence, « le téléphone sonne et il est dans la cuisine et il y a de la musique, ou bien Joe est dehors et le téléphone sonne et celui-ci est dans la cuisine») ; (b) en bas, sur fond vert, la spécification des actions à effectuer quand la situation se présente (dans l'exemple, « augmenter le volume du téléphone qui se trouve dans la cuisine »). Le menu circulaire (en bas à droite) regroupe les commandes du système. En particulier, la commande « prototype » lance l'exécution du programme en vue de sa mise au point.

Alors que la finalité de a CAPpella est d'aider le système à apprendre des situations à partir de données et d'événements de très bas niveau d'abstraction (signal son, vidéo, etc.), et à réagir à ces situations lorsqu'elles se présentent, iCAP : (a) s'appuie sur des concepts de plus haut niveau d'abstraction (concept de personne, d'ami, d'objet, de lieu, etc.) et (b) s'en tient à ce qui est spécifié par l'utilisateur. Autrement dit, il n'y a pas d'apprentissage, pas de généralisation dans iCAP. Il s'agit de programmation par *spécification* soit ex nihilo, soit à partir d'un programme existant, le *code généré de manière classique n'étant pas accessible* à l'utilisateur.

iCAP, comme a CAPpella, utilise la *manipulation directe WIMP classique* pour l'édition de programme avec les dispositifs usuels écran/souris/clavier. Comme le montre la figure III-9, le langage iCAP utilise les deux dimensions du plan pour traduire la sémantique des opérateurs logiques, de même pour les parties droite et gauche des règles. Il s'agit donc d'une *modalité de représentation graphique* (dite *langage visuel*)

qui n'est pas sans rappeler celle des AgentSheets[30] (Repenning 1993). Certains éléments de *stylistique* sont personnalisables comme les icônes qui servent de représentation des entités métier ; d'autres ne le sont pas, comme les fonds beige et vert qui permettent de distinguer les parties gauche et droite des règles. En l'état, iCAP est *monosyntaxe*, mais les auteurs envisagent une représentation textuelle des règles. Par construction, toute règle est *lexicalement* et *syntaxiquement correcte*. De plus, iCAP détecte les ambiguïtés au sein d'une règle, de même, les conflits entre règles : il y a donc support à l'*auto-correction sémantique*.

Les entités (concepts) métier sont répertoriées dans une base de connaissances éditable. Par exemple, iCAP permet au moyen d'un formulaire d'ajouter à la base de connaissances une nouvelle personne en y associant son nom, un dessin ou une photo, ou encore un nouveau dispositif pouvant servir de sortie (par exemple, la sonnerie d'un téléphone). La base de connaissances étant extensible, le langage est *extensible du lexique à la sémantique*.

Concernant sa couverture fonctionnelle, iCAP s'appuie sur la *réutilisation des services* et des connaissances maintenus par l'infrastructure, et permet la *réutilisation de programmes iCAP* (cf. la commande « file » du menu circulaire). Toutefois, iCAP ne fournit pas de service de recherche de programmes ni de gestion de versions. Il assure le *déploiement* grâce à l'infrastructure d'exécution.

Les possibilités de mise au point sont flexibles puisque trois modes sont offerts : soit l'utilisateur simule les données contextuelles (selon la technique usuelle du Magicien d'Oz) et la mise au point se fait dans le *monde numérique*, soit l'utilisateur branche le gestionnaire de contexte implémenté en l'occurrence avec le Context Toolkit (Salber et al. 1999) et la mise au point a lieu dans le *monde physique*, soit le gestionnaire de contexte est partiel et complété par simulation des données contextuelles manquantes : la mise au point est alors *mixte*. De ces trois modes de fonctionnement, nous en déduisons une *séparation mixte* entre les phases de conception et d'exécution.

Au bilan, iCAP est un environnement DUF assez complet quand bien même le problème de la qualité logicielle n'est pas abordé.

[30] Informations plus récentes disponible sur le site http://www.agentsheets.com/products

III-3 CAMP

CAMP (Capture and Access Magnetic Poetry) vise la même finalité qu'iCAP : permettre à des utilisateurs *constructeurs* de programmer des applications sensibles au contexte, mais cette fois ciblées sur l'*habitat intelligent* (Truong et al. 2004).

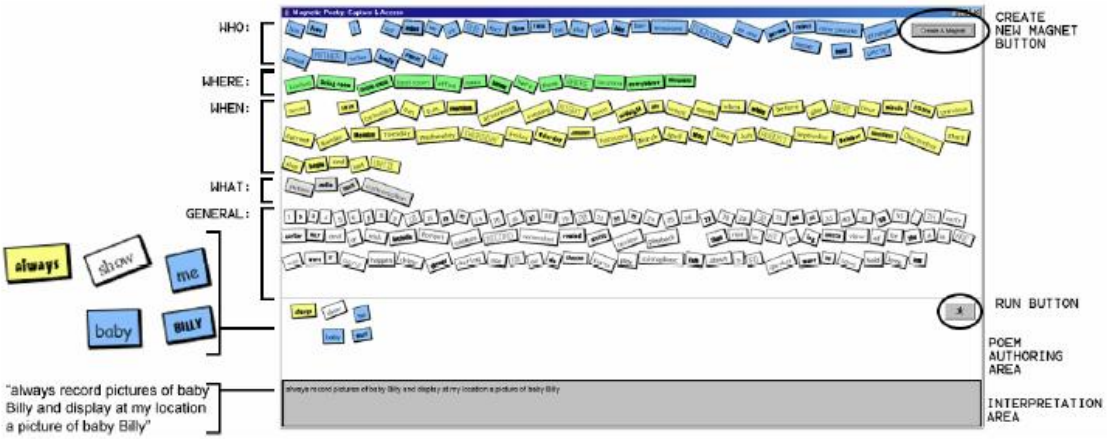


Figure III-10. L'IHM de l'environnement CAMP. L'écran est structuré en trois zones horizontales. La zone du haut montre le vocabulaire organisé par questions quintiliennes (Qui ? Où ? Quand ? Quoi ? etc.). Au centre, la phrase en cours d'édition (en l'occurrence, « Always, show me baby Billy ») construite par drag-and-drop à partir des mots du vocabulaire. Lorsque l'utilisateur appuie sur le bouton « Run », la phrase est traduite en une version compréhensible à la fois par le système et par l'utilisateur. Cette phrase, accessible en lecture, est affichée dans la zone du bas : « Always record pictures of baby Billy and display at my location a picture of baby Billy ». En haut, à droite, un bouton permet à l'utilisateur d'introduire un nouveau mot qui encapsule le programme constitué de la phrase courante.

Comme pour iCAP, il n'y a *pas de support au co-développement*, mais une attention particulière pour *une correspondance directe avec les tâches utilisateurs* et notamment des possibilités d'*expression de relations temporelles* (vraisemblablement tous les opérateurs d'Allen). Rappelons que la méthode de la « bande dessinée » évoquée au chapitre précédent (section I-1), vient de l'étude amont de CAMP. Parmi les résultats marquants de cette étude de terrain, Truong et al. remarquent que la plupart des participants mentionne rarement les dispositifs (caméras, capteurs, etc.), mais s'expriment en termes de fonctions[31]. Ce résultat n'est pas surprenant sachant que les dispositifs d'intérêt dans CAMP sont des dispositifs de capture, non pas, comme dans DisQo, les objets du quotidien. Du coup, contrairement à SpeakEasy (Edwards et al. 2001) dont la justification est la configuration de dispositifs, CAMP (comme iCAP) favorise le modèle « Situation-> Actions » et donc un paradigme de programmation *déclaratif par règle*. Il s'agit de *programmation ex nihilo par spécification*, le code généré étant *accessible en lecture* seule. Cette présentation du code sous forme d'une

[31] Un extrait représentatif qui illustre l'absence de référence aux dispositifs de capture au profit des fonctions attendues [tiré de (Truong, Huang, and G.D. Abowd 2004)] :"I am not very experienced in cooking, so I would want to record friends or relatives cooking [in my kitchen]. I would not have to take notes and I would be able to see and hear, step by step, how to make a particular dish. I would want the house to start recording when I told it to, and to stop when I told it to. Then I could review it and literally SEE [what to do while cooking]."

phrase en langage pseudo-naturel (cf. partie basse de l'écran de la figure III-10) constitue la seule aide à la *mise au point* que l'on peut considérer comme *quasi-absente* : l'utilisateur peut seulement vérifier *a priori* la conformité de la sémantique de son programme avec celle comprise par le système. Un bouton « run » permet de passer en mode exécution dans le monde réel. On observe donc une *séparation nette* entre les phases de conception et d'exécution mais le *déploiement est automatiquement assuré* par l'infrastructure d'exécution INCA (Truong et al. 2004).

La Figure III-10 montre l'IHM de l'environnement CAMP. Le vocabulaire et la syntaxe sont ceux d'un *langage pseudo-naturel* c'est-à-dire un langage naturel contraint. La modalité de représentation est *textuelle* avec rendu sur écran. La construction de phrase s'effectue par *manipulation directe* classique. Des effets de *stylistiques* sont exploités pour améliorer la lisibilité et la facilité d'écriture : (1) à chaque catégorie de mots est associée une couleur (par exemple, le jaune pour les mots intervenant dans l'expression du temps) ; (2) chaque mot est entouré d'un rectangle figuratif par analogie avec les aimants que les familles (américaines surtout) ont l'habitude de placer sur leur réfrigérateur. On peut aisément imaginer une version tangible du même langage où chaque mot est inscrit sur un aimant physique à la manière du Senseboard de Jacob et al. (Jacob et al. 2002). Le langage est *extensible du lexique à la sémantique* : l'utilisateur peut créer de nouveaux mots (cf. bouton en haut et à droite de l'écran de la figure III-10). Chaque mot ainsi créé correspond à l'encapsulation de la phrase en cours d'édition à la manière d'une macro sans paramètre. On peut voir dans cette facilité un début de réutilisabilité, mais aussi un moyen de structuration élémentaire du langage *en niveaux de complexité*.

III-4 AutoHAN

AutoHAN est une infrastructure d'exécution spécialement dédiée à la spécification, par l'utilisateur final, des interactions entre dispositifs domestiques (télévision, lumières, mais aussi dispositifs d'interaction tels les Media Cubes introduits en section II-3-5) (Blackwell et al. 2002). Le domaine visé est donc l'*habitat intelligent*. Contrairement à CAMP qui part des tâches utilisateur, autoHAN est centrée sur les dispositifs mais en exploitant l'expérience que les gens ont acquise avec les télécommandes.

Au-delà de sa capacité à masquer l'hétérogénéité des réseaux (Infrarouge, ATM, Ethernet, BlueTooth) comme celle des dispositifs via le protocole UPnP, l'élément distinctif d'autoHAN est d'être *multisyntaxe* (ou plus exactement ici, multilangage) Comme le montre la figure III-11, tout programme spécifié dans un langage donné est traduit dans un langage pivot appelé opportunément Lingua Franca.

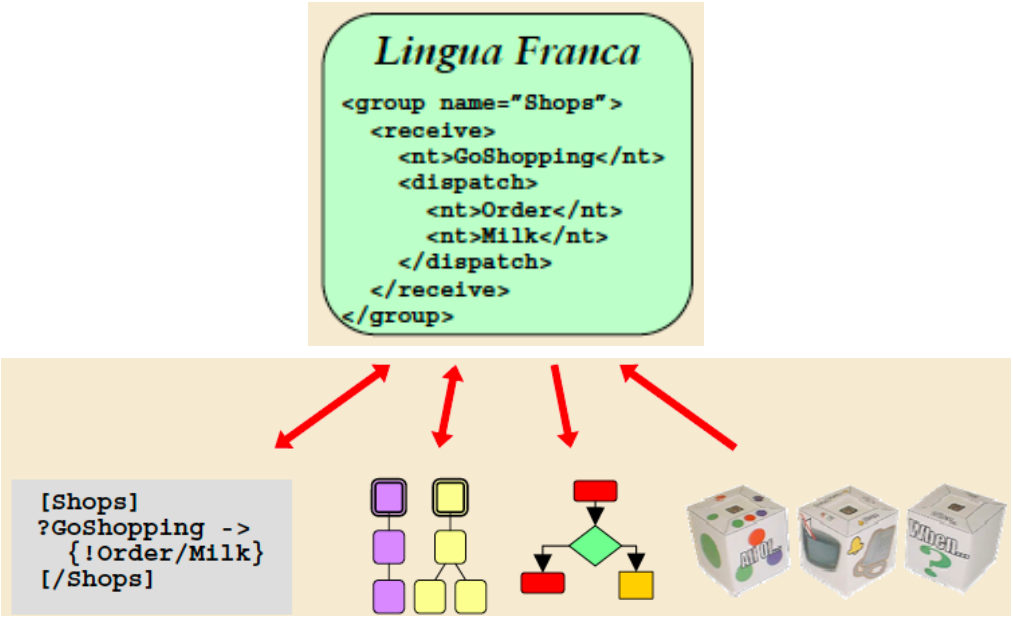


Figure III-11. AutoHAN et sa *Lingua Franca* comme langage pivot entre plusieurs syntaxes. À gauche, une syntaxe de représentation textuelle, au centre une modalité de représentation graphique de type langage visuel, à droite les media cubes comme supports physiques de rendu des termes du langage. Les flèches traduisent le sens des traductions possibles entre langages.

Lingua Franca obéit au *paradigme de programmation par événements* facilement implémentable avec le protocole GENA sur lequel autoHan s'appuie. Le langage visuel Vseq de la figure III-12 est parfaitement calqué sur ce paradigme. Ainsi, en théorie, l'utilisateur final, qui est un *constructeur*, peut éditer ses programmes dans la(les) notation(s) de son choix. Toutefois, l'*égale opportunité* entre les syntaxes (ici, les langages) *est partielle* : comme le montre les flèches de la figure III-11, 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, il n'est pas possible 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 exprimé en Vseq ! En outre, la couverture sémantique d'un langage PUF d'autoHAN est au mieux celle de Lingua Franca. Autrement dit, chaque langage est censé s'adresser à des utilisateurs finaux dont les compétences et les exigences en matière de programmation peuvent aller du débutant (cas des Media Cubes) à l'expert (cas de Vseq). Cette multiplicité de langages PUF est une façon de structurer l'environnement en *niveaux de complexité*.

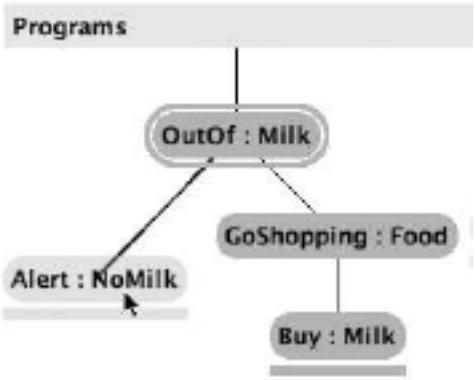


Figure III-12. Un programme dans la syntaxe du langage VSeq disponible dans autoHAN [d'après (Hague et al. 1997)].

La programmation avec les Media Cubes est originale. Chaque face de cube, on l'a vu, sert de support à une unité syntaxique du langage. Les cubes sont typés et sont liés dynamiquement à des entités de la maison en jouant essentiellement sur la proximité spatiale. Par exemple, pointer la télécommande de la télévision vers le cube « time » suivi d'un appui sur la touche 1 de cette télécommande, correspond à la construction d'un programme qui va lancer la 1ère chaîne à cette même heure le lendemain et cela, une seule fois. Ou encore, pour construire un radio-réveil, il suffit de deux cubes : la face « do » d'un cube (qui représente une expression conditionnelle) est liée à une représentation de l'action « allumer la radio » du monde réel – cette représentation peut être la radio elle-même ou son bouton de mise en marche ; la face « when » du second cube (qui représente le temps) est liée à une représentation, dans le monde réel, de l'heure souhaitée – cette représentation peut être une horloge.

Les couplages « face de cube – entités du monde réel » traduisent des affectations de variable, ce qui offre une grande souplesse d'expression. L'utilisateur ne construit pas un modèle physique d'un programme, mais construit un programme par manipulation de cubes physiques, manipulation fortement inspirée de celle des télécommandes. Toutefois, sans la représentation équivalente dans une autre syntaxe sur support numérique, l'utilisateur n'aurait aucun moyen de se souvenir de la sémantique des couplages des cubes, ni de réutiliser ses programmes, ni de les tester autrement que dans le monde physique.

Comme pour a CAPpella, iCAP et CAMP, il n'y a *pas de support au co-développement*. Il s'agit de *programmation* essentiellement *ex nihilo par spécification*, le code généré étant *accessible en lecture* comme *en écriture* : Lingua Franca étant exprimée en XML, il est possible de modifier ce code directement au risque de perdre la conformité avec le programme source en l'absence de transformation inverse. En l'état, autoHAN permet la *réutilisation basique* de programme sans gestion de version et l'aide à la *mise au point est quasi-inexistante* si ce n'est la capacité de consulter son code dans une autre notation et de l'exécuter dans le *monde physique*. On observe une *séparation nette* entre les phases de conception et d'exécution, mais le *déploiement est automatiquement assuré* par autoHAN.

III-5 BYOB appliqué aux dispositifs domestiques

BYOB (Build Your Own Blocks) est une extension du langage Scratch et de son environnement. BYOP permet de construire des macro paramétrées appelées « blocks ». Extension de Scratch, il en hérite les caractéristiques (Ash et al. 2011). Il s'agit en effet d'un *langage impératif* dont la syntaxe relève des *langages visuels* (cf. section II-3-5) avec la *stylistique* de Scratch. L'éditeur BYOB permet de définir de nouveaux blocks par *manipulation directe*. Il suffit donc de construire des blocks adaptés au domaine cible, ces blocks servant alors de primitives pour programmer le domaine cible, mais à plus haut niveau d'abstraction que Scratch. BYOB est donc *extensible, du lexique à la sémantique*, mais il est *monosyntaxe* (héritage de la syntaxe de Scratch).

Par exemple, la figure III-13 montre deux classes de blocks dédiés au contrôle d'appareils domestiques : à gauche, les primitives d'acquisition des données d'entrée (l'heure, données fournies par les capteurs environnementaux, actions humaines, état des dispositifs) ; à droite les primitives de sortie, et notamment l'expression de changement d'état des dispositifs ; au centre, les primitives natives de Scratch. Les blocks de la figure III-14 montrent comment lancer la machine à café à 7h57 du matin, puis faire sonner le réveil-matin, et jouer une pièce musicale lorsque le café est prêt. On trouvera dans (Ash et al. 2011) d'autres exemples de blocks conçus pour la programmation d'appareils domestiques fondés sur BYOB.

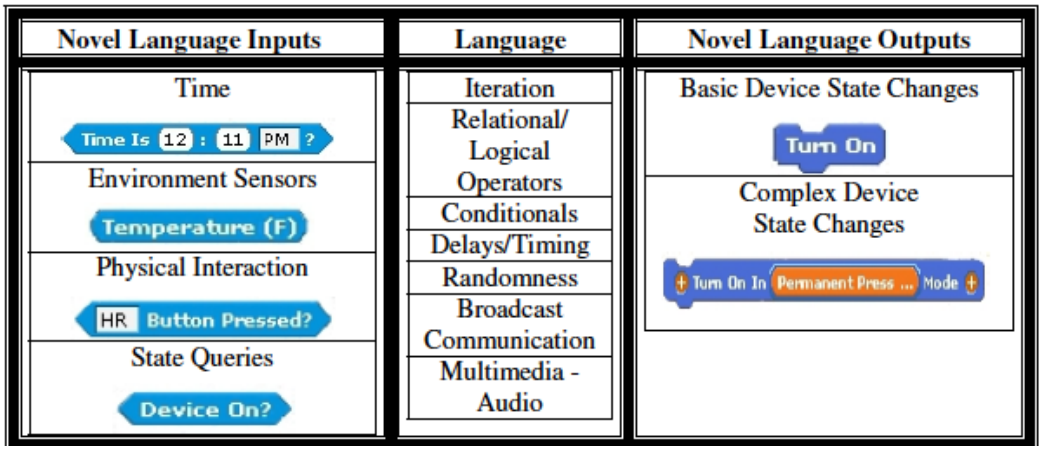


Figure III-13. Exemple de primitives pour le contrôle de dispositifs domestiques construites avec BYOB où chaque primitive est un « Block » BYOB et où chaque dispositif est un « sprite » Scratch (Ash et al. 2011).

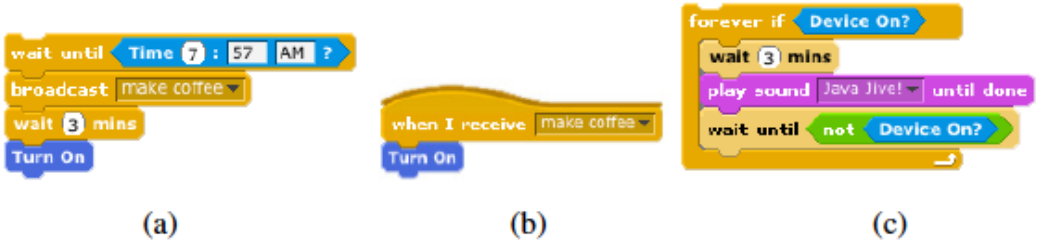


Figure III-14. Exemple de blocks dédiés à la programmation de dispositifs domestiques. Le block (a) est exécuté par le « sprite » réveil-matin, les blocs (b) et (c) par la machine à café. a) Le réveil-matin attend 7h57 pour émettre le message « make coffee ». Il attend trois minutes en sorte que le réveil sonne à 8h00. b) La machine à café se met en route dès la réception d'un message « make coffee ». c) Chaque fois que la machine à café se met en route, elle annonce lorsque le café est prêt (Ash et al. 2011).

À l'évidence, l'utilisateur est un *constructeur* : il *programme* son habitat en réutilisant et en paramétrant des blocks BYOB (que l'on peut considérer comme des *exemples*), mais aussi en *codant* avec les *primitives Scratch*, ce qui offre une bonne puissance d'expression. Par conséquent, il est possible, en théorie, *de couvrir les opérateurs d'Allen*. Le programme est traduit dans Processing[32] qui inclut à la fois un langage basé Java et un environnement d'exécution. Le programme compilé n'est pas accessible à l'utilisateur. Processing sert d'intergiciel entre les programmes BYOP et le méga microcontrôleur Arduino auquel sont connectés les dispositifs domestiques.

Si les blocks dédiés métier sont censés offrir une *correspondance directe*, il n'est *pas certain que l'environnement BYOB soit structuré en niveaux de complexité*. Le support à la mise au point est quasiment absent : l'utilisateur peut voir l'état des appareils domestiques représentés par des icônes dans une fenêtre dédiée. Rien n'est dit si le programme s'exécute en même temps dans le monde réel.

Au bilan, l'application de BYOB à la programmation de dispositifs domestiques ouvre une voie intéressante, sachant que Scratch a été conçu pour l'apprentissage de la programmation par des enfants. Toutefois, ces développements avec BYOB sont des applications jouets : l'infrastructure d'exécution ne passe pas à l'échelle et aucun test d'utilisabilité n'a été conduit.

Nous avons jusqu'ici analysé les contributions de la communauté IHM que nous estimons les plus représentatives. Nous étudions maintenant quelques-unes émanant de la communauté systèmes distribués et intergiciel.

III-6 TeC

TeC est une infrastructure d'exécution et de déploiement orientée services qui permet à un utilisateur final *constructeur* de fabriquer ses propres espaces intelligents (Sousa et al. 2011). TeC s'appuie sur les concepts d'acteurs (player), d'activités et d'équipe (team). Un acteur est un dispositif matériel doté de capacité de calcul et de

[32] http://www.processing.org

communication (par exemple, un téléphone, un capteur ou encore un appareil électroménager). Une activité généralise, sous une même bannière, les concepts de processus, de service de calcul, ou de fonction de dispositifs. Au moyen d'un éditeur, l'utilisateur final construit à façon des équipes par *composition* d'acteurs pour assurer le service qui lui convient. Par exemple, la figure III-15 montre une équipe de surveillance composée de trois acteurs (monitor fence, phone et film) qui échangent, via des chemins établis par l'utilisateur final, des événements d'entrée et de sortie dont la nature et les conditions d'occurrence sont également spécifiées par l'utilisateur. Dans cet exemple, lorsque la barrière détecte une condition d'alarme, la scène est filmée et le propriétaire en est averti sur son téléphone avec le message « There's a possible break in the perimeter » accompagné de la scène distante filmée en temps réel. Le flux vidéo est arrêté dès réception d'un accusé de réception émis au moyen du téléphone.

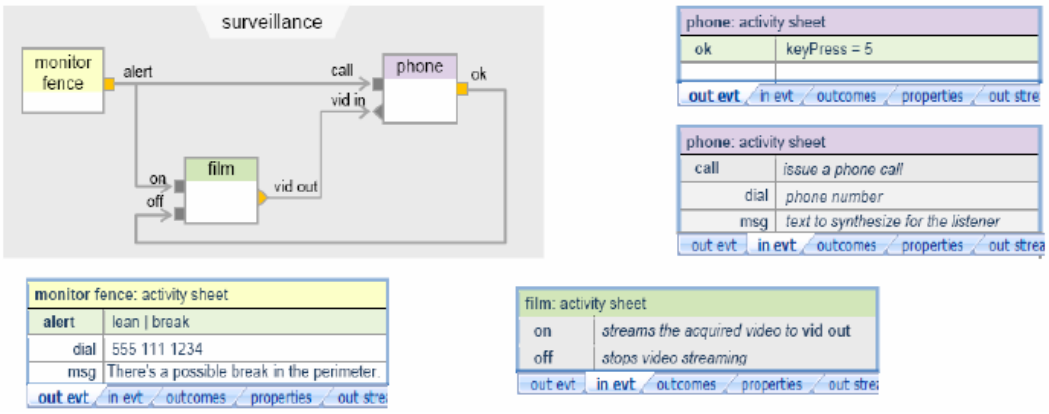


Figure III-15. Les principes de TeC. En haut à gauche, l'équipe *surveillance* est une composition de trois acteurs interconnectés que l'utilisateur a construite par spécification. Le comportement de ces acteurs est spécifié dans des feuilles d'activité (*activity sheets*) dont la syntaxe est inspirée de celle des tableurs. En bas à gauche, la feuille d'activité de *monitor fence* qui produit un événement *alert* (onglet *out evt* de la feuille d'activité) lorsque l'une des conditions *lean* ou *break* est remplie. *alert* comprend deux paramètres : *dial* (la valeur du numéro de téléphone à composer) et *msg* avec sa valeur. L'événement *alert* est reçu par les acteurs *film* (via le chemin qui relie *alert* à *on*) et *phone* (via le chemin qui relie *alert* à *call*). L'événement d'entrée *call* de l'acteur *phone* (onglet *in evt*), au centre à droite, indique la nature des champs attendus (*dial* et *msg*) qui se trouvent être conformes au contenu de l'événement reçu (*alert*). Inversement, le format d'*alert* n'est pas conforme à l'événement d'entrée *on* de l'acteur *film*. Dans ce cas, les paramètres *d'alert* sont ignorés, mais pas son occurrence ce qui suffit à déclencher l'activité d'enregistrement vidéo : TeC effectue les alignements de conformité sur les noms selon l'algorithme du « best effort ». En haut à droite, la pression de la touche 5 du téléphone (*out evt ok*) suffit à provoquer l'arrêt de l'enregistrement vidéo.

Le langage TeC permet de faire référence à des informations contextuelles telles les expressions de localisation et de relations spatiales de la figure III-16. L'infrastructure prend alors en charge de ne faire intervenir que les acteurs répondant à ces critères. A contrario, les *opérateurs d'Allen ne sont pas couverts* de manière explicite, chaque acteur réagissant par causalité aux événements reçus.

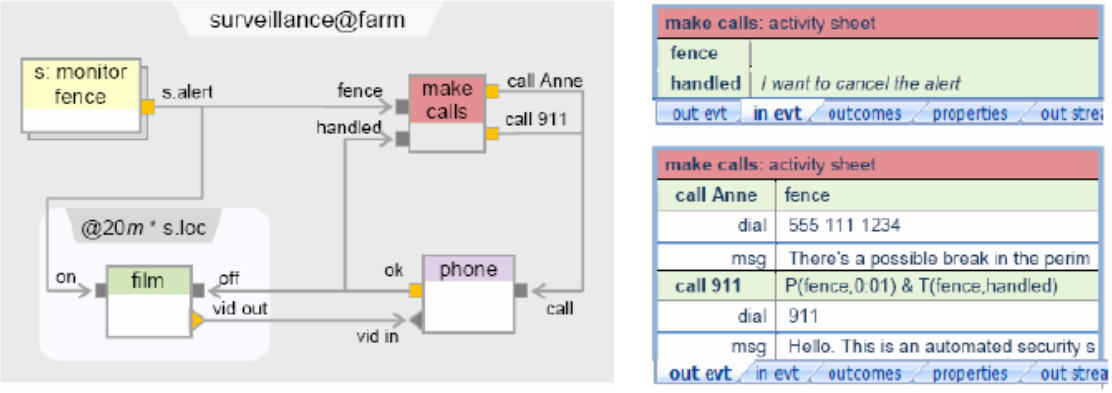


Figure III-16. TeC et la notation @ pour désigner des informations contextuelles spatiales. Dans cet exemple, tous les acteurs de l'équipe *surveillance* sont situés physiquement dans le lieu symbolique *farm*. la variable *s* désigne une instance d'acteur *monitor fence* pris dans l'ensemble des acteurs de type *monitor fence* situés dans *farm*. L'expression spatiale *@20m*s.loc* désigne la caméra située à 20 mètres de la localisation de *s* émetteur de l'alerte. Noter dans la feuille d'activité de *make calls*, la condition d'envoi de l'événement *call 911* : l'appel du 911 n'aura lieu que si l'utilisateur n'a pas accusé réception de l'alerte au bout d'une minute (*P* pour *Postpone* et *T* pour *Toggle*). Pas sûr qu'un utilisateur final soit capable de spécifier une telle condition !

Les Figures III-17 et III-18 montrent un exemple concret d'utilisation de l'éditeur TeC pour construire, sur Android, l'équipe de surveillance de la figure III-15. L'expression de l'adaptation, nous l'avons vu, se fait *par composition* d'acteurs, c'est-à-dire de composants pré-existants, et cela de manière *déclarative* selon un *modèle par événements* : l'utilisateur final déclare les acteurs, les connexions, les conditions d'occurrence des événements transitant sur ces connexions, et l'encapsulation dans des équipes. L'*extensibilité* est de nature *sémantique* à deux niveaux : (1) l'infrastructure TeC, qui découvre de manière automatique les acteurs de l'espace intelligent, rend possible l'ajout de nouveaux acteurs, donc de nouvelles fonctions. (2) L'utilisateur définit les connexions entre acteurs et donc de nouveaux comportements.

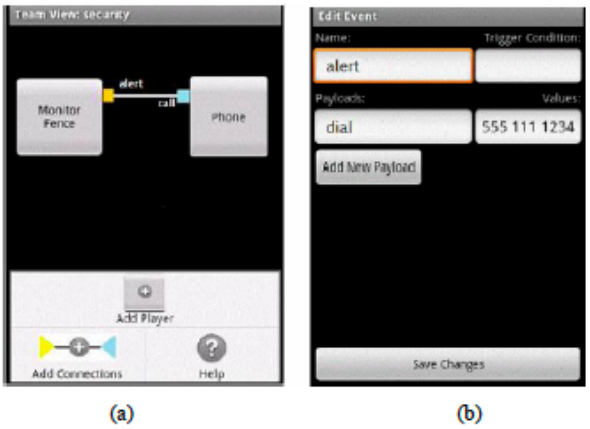


Figure III-17. Editeur TeC sur Android. (a) En bas de l'écran, les commandes pour ajouter un acteur et établir des connexions entre acteurs. (b) Spécification des paramètres des événements.

Le langage TeC est *monosyntaxe*, mais il utilise *deux modalités de représentation graphique* selon des *techniques d'interaction WIMP* ultra-classiques (absence totale de manipulation directe !) : d'une part, des graphes pour représenter les acteurs et leurs interconnexions, d'autre part, des formulaires inspirés des tableurs pour les spécifications des ports de connexion et les conditions d'occurrence des événements.

Figure III-18. (a) Ajout de l'acteur film à la configuration de la figure III-16. (b) Ajout d'une connexion de type *stream* entre *film* et *phone*. (c) Désignation des noms des ports de sortie (*vid out*) et d'entrée (*vid in*) de la connexion entre *film* et *phone*.

Par sa capacité à découvrir les acteurs sur critère, TeC assure *de facto* un support à la *réutilisation*, mais aussi à la *correction sémantique*. L'utilisateur peut réutiliser un acteur tel quel, réduisant ainsi le coût cognitif d'entrée. Il doit néanmoins spécifier les connexions entre les acteurs d'une équipe. Nous voyons là une *faille dans le principe de structuration en niveaux de complexité*. L'encapsulation en équipe définit un niveau d'abstraction plus approprié, sous réserve que l'environnement TeC fournisse les moyens de gérer les équipes. Bien que TeC suppose l'existence d'une sorte d'appStore d'acteurs et d'équipes, il n'existe *pas d'aide explicite au co-développement*. En l'état, TeC n'inclut *pas d'aide à la mise au point*, mais grâce à son infrastructure d'exécution dynamique (voir figure III-19), *les phases de conception et d'exécution sont fortement couplées*.

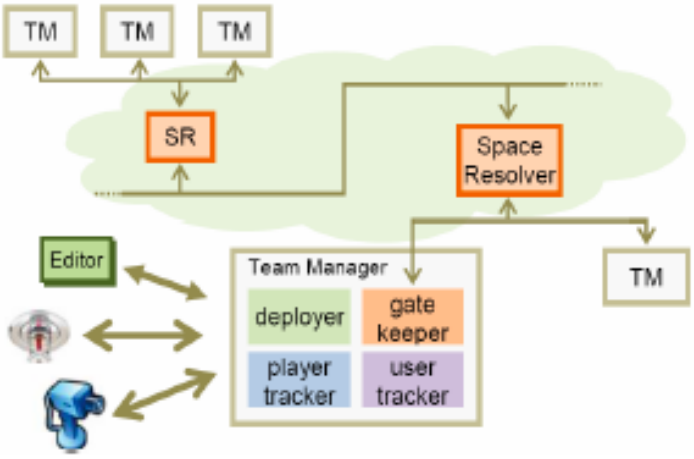


Figure III-19. Décomposition fonctionnelle de l'infrastructure d'exécution TeC. Les flèches représentent la communication d'événements TeC (exprimés en XML) au-dessus de TCP-IP. Une équipe d'acteurs est gérée par un Team Manager (TM). Les acteurs sont supposés être physiquement colocalisés. Un TM inclut : (1) un service d'enregistrement et de suivi de la localisation et de la performance des acteurs de l'équipe (player tracker) ; (2) un service de déploiement des acteurs de l'équipe (deployer) ; (3) un service de suivi de la localisation physique de l'utilisateur (user tracker) ; (4) un service de communication avec les autres TM. Les Space Resolvers (SR) servent essentiellement à identifier les acteurs répondant aux contraintes spatiales exprimées par l'utilisateur final (cf. *@farm* et *@20m*s.loc* de la figure III-16). *Editor* désigne l'outil de construction d'équipes TeC.

III-7 PANTAGRUEL

Comme TeC, Pantagruel s'adresse à un utilisateur *constructeur*. Comme TeC, Pantagruel est largement influencé par le modèle à composants des systèmes distribués (Drey 2010). Toutefois, l'utilisateur final « voit » des objets (au sens de l'approche à objets). Alors que dans TeC l'utilisateur spécifie explicitement les connexions entre les composants, Pantagruel construit ces connexions à partir de la spécification de *règles d'orchestration* entre objets. Comme le montre la figure III-20, la partie gauche d'une règle, ou condition, fait référence à des informations issues de capteurs (logiques) reliées par des opérateurs logiques (et, ou). La partie droite fait référence à des effecteurs (logiques). Capteurs et effecteurs sont des composants (entités objets) de l'infrastructure. Notons que ces règles d'orchestration, qui spécifient les conditions d'échange entre les objets, sont fonctionnellement équivalentes aux feuilles d'activité de TeC.

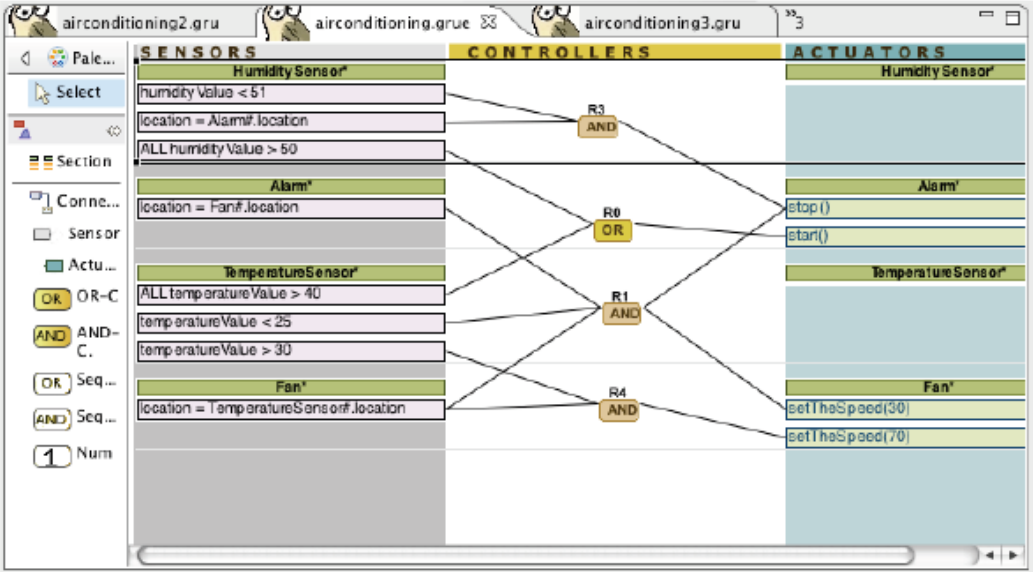


Figure III-20. Spécification Pantagruel de règles d'orchestration pour le confort domestique. A gauche, les unités lexicales et syntaxiques du langage. A droite, la zone d'édition de programmes Pantagruel. La zone d'édition est structurée en trois colonnes : sensors, controllers, actuators. La règle R3 stipule ceci : « Si la valeur du capteur d'humidité est inférieur à 51 et si ce capteur est au même endroit que l'alarme alors arrêter l'alarme ». Avec R0, on démarre l'alarme si tous les capteurs d'humidité indiquent une valeur supérieure à 50 (probablement s'agit-il de pourcentage) ou si toutes les températures sont supérieures à 40 (probablement s'agit-il de degré Celsius).

Pantagruel est un outil *général paramétré* (comme notre outil KISS décrit au chapitre suivant) : une description sous forme de diagrammes de classe UML du domaine cible est utilisée en entrée pour produire un éditeur Pantagruel *spécifique*. La Figure III-21 montre l'ontologie correspondant à la fenêtre d'édition de la Figure III-20. Celle-ci est accessible à l'utilisateur final. Pantagruel offre donc les moyens techniques d'assurer une *correspondance directe* avec le domaine cible mais *sans toutefois proposer de structuration en niveaux de complexité* : si Pantagruel masque l'expression explicite des connexions entre les composants de l'espace intelligent, l'utilisateur se doit de comprendre un diagramme de classe UML, la notation pointée ou encore les symboles * et # pour désigner un ensemble ou une instance de composant.

Concernant la lisibilité et la facilité d'écriture, Pantagruel est *monosyntaxe* et relève des *langages visuels* : il utilise la notation graphique des graphes ainsi que trois colonnes pour distinguer les rôles des unités syntaxiques. La *stylistique* inclut une couleur de fond spécifique à chaque colonne, de même pour les noms des composants ou des opérateurs. Un effort particulier doit être relevé concernant le support à la *correction syntaxique et sémantique* : d'une part, les conditions ne peuvent se construire qu'à partir des unités syntaxiques issues de l'ontologie ; d'autre part, tout programme Pantagruel peut être traduit en TLA+ et de là, avec le model checker TLC, conduire une campagne de vérification formelle de propriétés clefs du système.

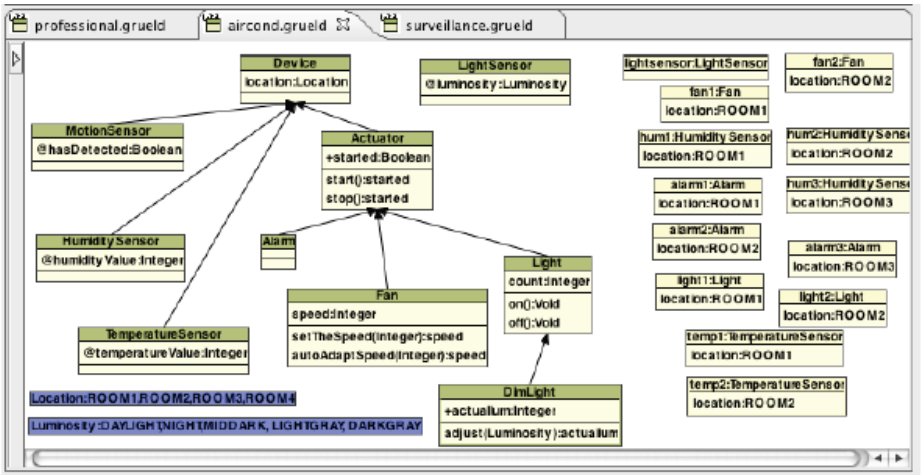


Figure III-21. Ontologie UML pour le confort domestique utilisée comme paramètre d'entrée de l'éditeur de la Figure III-20.

L'expression de l'adaptation est par *composition* (implicite) de composants dont l'orchestration est *programmée par spécification de règles*. Ces règles sont ensuite *traduites* dans l'ADL DiaSpec (*génération classique de code*) (Jouve et al. 2008), code *non accessible* à l'utilisateur final mais qui peut être déployé dans le monde réel ou servir d'entrée au simulateur, DiaSim. DiaSim sert de metteur au point dans le monde numérique (voir Figure III-22). Le support à la *mise au point est donc mixte*.

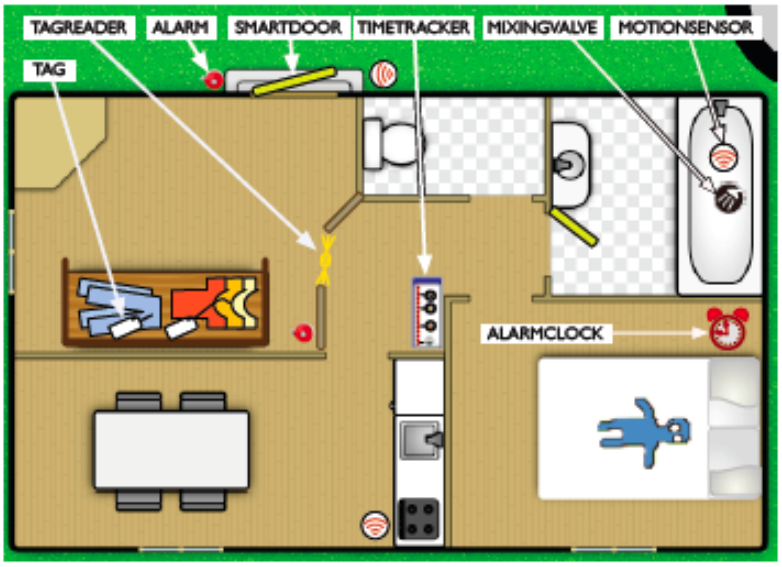


Figure III-22. DiaSim sert de metteur au point de programmes Pantagruel dans un monde simulé.

Le paramétrage (réalisé une fois, par métier cible), puis le processus de compilation utilisés par Pantagruel (voir figure III-23), indiquent qu'il y a, en l'état, *séparation stricte entre phases de conception et phase d'exécution* : comme indiqué plus haut, les règles d'orchestration ne peuvent faire référence qu'aux entités (composants) d'une ontologie (taxonomie) déclarée en UML et compilée en Java par le compilateur DiaSpec pour constituer un « framework ». Les règles d'orchestration d'un programme

Pantagruel sont, elles, compilées dans l'ADL de DiaSpec. L'ensemble, framework et code ADL, est ensuite déployé sur la plate-forme d'exécution idoine : celle du monde réel ou DiaSim.

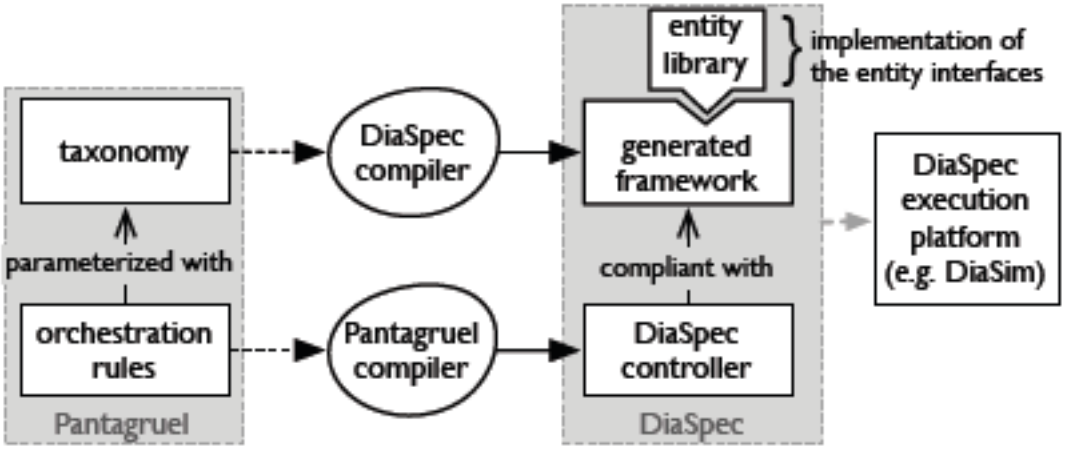


Figure III-23. Processus de compilation de Pantagruel.

IV – Analyse synthétique de l'état de l'art

Les tableaux des figures III-24 et III-25 situent l'ensemble des propositions de l'état de l'art que nous avons retenues sur la programmation de l'habitat intelligent. L'analyse de ce tableau appelle les remarques suivantes présentées selon les trois grandes dimensions de notre espace problème.

				A Cappella	iCAP	CAMP	AutoHAN	BYOB	TeC	Pantagruel
Utilisateur Final	Consommateur									
	Délégateur									
	Constructeur			X	X	X	X	X	X	X
Environnement de développement et d'exécution	Couverture fonctionnelle	Expression des requis								
		Conception et spécification								
		Programmation		X	X	X	X	X	X	X
		Réutilisation		X	X	X	X	X	X	X
		Maintenance								
		Gestion de versions								
		Mise au point	Monde numérique uniquement							
			Monde physique uniquement			X	X	X	X	
			Mixte	X	X					X
	Support au co-développement	Espace-temps	À distance							
			Colocalisation							
			Prévisible							
			Synchrone							
			Asynchrone							
			Prévisible							
		Activités collaboratives	Coproduction							
			Coordination							
			Communication							

	Couplage entre phase de conception et phase d'exécution	Séparation forte	X		X	X			X
		Phases confondues						X	
		Séparation mixte		X					

Figure III-24. Classement des propositions de l'état de l'art dans les dimensions « Utilisateur final » et « Environnement de développement ».

					A Cappella	iCAP	CAMP	AutoHAN	BYOB	TeC	Pantagruel
Programmation et Edition	Lisibilité et facilité d'écriture	Modalité de représentation (en sortie)		Dispositif	w	w	w	w	w	w	w
				Langage de représentation	lv	lv	T	ms	lv	lv	lv
		Technique d'interaction (en entrée)			md	md	md	ms	md	md	md
		Stylistique et sucre syntaxique				X	X	X	X	X	X
		Approche multisyntaxe		Egale opportunité							
				Egale opportunité partielle				X			
				Pas d'égale opportunité							
		Support à la correction		Correction sémantique	X	X			X	X	X
				Correction syntaxique	X	X	X		X	X	X
				Correction lexicale		X	X	X	X	X	X
	Expressivité de l'adaptation	Par paramétrage									X
		Par composition								X	X
		Par extension			X						
		Par programmation	Ex nihilo			X	X	X			
			À partir d'exemples		X	X			X		
			Par spécification	Code généré accessible (lecture, écriture)			\ lo	X			
				Code généré inaccessible	X	X			X		X
				Génération par compilation							X
				Génération par généralisation	X						
			Par codage						X		

Paradigme	Impératif					X		
	Déclaratif			X			X	
	par objets							X
	par règles	X	X	X				X
	par événements				X		X	
	par aspects							
	Multi-paradigme							
Extensibilité	Extensibilité lexicale	X	X	X		X	X	X
	Extensibilité syntaxique	X	X	X		X	X	X
	Extensibilité sémantique	X	X	X		X	X	X
Spécificité	Structuration en niveaux de complexité			X	X			
	Correspondance directe	X	X	X	X	X	X	X
	Domaine d'application	sc	sc	hi	hi	hi	ei	ei

Figure III-25. Classement des propositions de l'état de l'art dans la dimension « Programmation et outils d'édition ». W = wimp, lv = langage visuel, md = manipulation direct, ms=multisyntaxe, T=textuel, lo =lecture seulement, sc = sensible au contexte, hi=habitat intelligent, ei = espace intelligent.

1- Concernant l'utilisateur cible, toutes les solutions s'adressent au constructeur.

2- Concernant l'environnement de développement et d'exécution :

- Aucune proposition ne couvre l'expression des requis ou le support au co-développement. En corollaire, la réutilisation n'est pas traitée de manière explicite. Elle est esquissée *a minima* dans iCAP via les commandes Open et Save offertes par les systèmes de fichier classiques. Nous sommes loin d'une « place de marché pour composants logiciels » !
- Les outils d'aide à la mise au point sont encore peu avancés. Les idées telles que la Whyline (Ko et al. 2004), le "What You See Is What You Test"[33] (WYSIWYT) appliqué au tableur par Fisher (Fisher et al. 2006), ou des "data description" de Scaffidi (Scaffidi 2009), n'ont pas été considérés.
- Nous constatons une séparation nette entre les phases de conception et d'exécution à l'exception des solutions qui s'appuient sur une infrastructure d'exécution à composants orientés services dynamiques (cas de TEC).

3- Concernant les langages de programmation et les outils d'édition :

[33]En français : « Ce que vous voyez est ce que vous testez ».

- Toutes les solutions sont spécifiques à un domaine. Il en résulte une correspondance directe avec les besoins de l'utilisateur final sans toutefois assurer une bonne couverture des opérateurs temporels d'Allen. A contrario, peu d'attention a été portée sur la nécessité de structurer les outils d'édition en niveaux de complexité. Notons que Pantagruel couvre à la fois la généralité et la spécificité en prenant comme paramètre une description de l'ontologie du domaine cible.
- La plupart des solutions s'appuie sur les langages visuels manipulables via des dispositifs d'interaction classiques (écran, souris, clavier). Les media blocks d'autoHAN font figure d'exception dans ce paysage WIMP usuel puisqu'ils sont les seuls à relever des IHM tangibles. Toutes les propositions font usage de traits stylistiques, mais seul iCAP en permet la personnalisation. Aucune n'envisage les représentations en 3D.
- Les éditeurs sont massivement monosyntaxes à l'exception d'AutoHAN qui n'assure toutefois que l'égale opportunité partielle. Tous satisfont le critère de la correction lexicale et syntaxique, voire sémantique. Mais seul Pantagruel permet d'effectuer des vérifications formelles de propriétés.
- Concernant les paradigmes de programmation, nous observons peu d'exemples d'approche impérative (BYOB). La tendance s'oriente clairement vers des approches déclaratives à base de règles ou d'événements.
- L'extensibilité est plutôt bien couverte à tous les niveaux d'abstraction.
- Concernant l'expression de l'adaptation, nous assistons principalement à une combinaison de procédés par composition et par programmation, aussi bien *ex nihilo* qu'à partir d'exemples. 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 au langage Lingua Franca qui lui sert de pivot. Notons aussi pour CAMP l'accès en lecture au code généré (en l'occurrence, une phrase en langage pseudo-naturel).

Dans KISS, nous traitons un sous-ensemble de ces limitations. C'est l'objet du chapitre qui suit.

Chapitre IV : KISS, un environnement de développement par l'utilisateur final, et son infrastructure d'exécution, CONTINUUM

Table des matières :

Avant-propos 86

I – CONTINUUM, l'infrastructure d'exécution 87
I-1 Principes directeurs de WCOMP 87
I-2 Architecture logicielle de CONTINUUM 92

II – Un exemple de méta-IHM réflexe 98
II-1 L'interaction dans PhotoBrowser 99
II-2 Mise en œuvre 100

III – KISS vu de l'utilisateur final 102
III-1. KISS dans l'espace problème PUF-DUF-GLUF 102
III-2 KISS en action 103

IV – Mise en œuvre de KISS 108
IV-1 Présentation générale 108
IV-2 Grammaire KISS du langage de programmation 110
IV-3 Analyseur LX-SX et la représentation pivot 112
IV-4 L'éditeur de programme EdProg 115
IV-5 Analyseur sémantique et génération de code 119
IV-6 Simulateur, metteur au point 129

V – Conclusion 130

Résumé :

Ce chapitre présente KISS (Knit your Ideas Into Smart Spaces), un outil DUF (Développement par l'Utilisateur Final), et son infrastructure d'exécution, CONTINUUM (CONTinuité de service en Informatique UbiqUitaire et Mobile). CONTINUUM traite du problème de l'hétérogénéité des dispositifs et des services numériques, ainsi que de l'adaptation dynamique aux changements d'état des mondes physiques et numériques. Elle s'appuie pour cela sur l'intergiciel WCOMP. Cet intergiciel implémente un modèle de service et de composant dont la composition et la reconfiguration dynamiques sont exprimées à l'extérieur de la « logique applicative » sous forme de plans d'adaptation exprimés en Aspect d'Assemblage (AA). KISS, qui se présente comme l'un des services de CONTINUUM, génère de tels plans.

KISS se caractérise comme suit : il s'adresse à un utilisateur final constructeur. Sa couverture fonctionnelle inclut la programmation, la mise au point et dans une moindre mesure, la réutilisation. Le langage de programmation est de type déclaratif orienté règles, avec égale opportunité syntaxique entre langue française pseudonaturelle (LPN) et langage visuel iconique. La technique d'interaction de construction des programmes LPN s'appuie sur l'utilisation de menus dont les options sont calculées dynamiquement à partir de l'état de l'habitat intelligent connu de CONTINUUM. KISS assure ainsi la découverte progressive du langage ainsi que l'extensibilité et la correction syntaxique et sémantique. Les programmes donnent lieu à une génération de plans d'adaptation exprimés en AAs et exécutés par CONTINUUM. La mise au point peut se pratiquer au choix, dans le monde physique, dans un monde dual numérique ou dans les deux simultanément[34]. KISS ne fournit aucun support au co-développement, jugé pour l'instant, peu prioritaire. Par contre, il est le seul, avec TeC, où phases de conception et d'exécution peuvent être perçues comme confondues. KISS se démarque toutefois de ce dernier par une IHM plus accessible aux non informaticiens.

[34] S'il est possible, par conception, d'exécuter un programme dans les deux mondes à la fois, la version actuelle de KISS ne permet l'exécution que dans un seul monde à la fois.

Avant-propos

Rappelons que l'étude préliminaire de terrain présentée au chapitre II nous a permis de vérifier que l'utilisateur final était disposé à programmer son habitat et notamment à coupler des dispositifs pour obtenir de nouveaux services. Forts de cette confirmation et en accord avec notre approche (cf. figure IV-1), nous avons analysé l'état de l'art, à la recherche d'outils PUF-DUF-GLUF idoines (chapitre précédent). Bien que l'offre soit riche, nous avons constaté bon nombre de lacunes : absence de support à l'expression des requis ou au développement collectif, peu d'aide à la mise au point, dichotomie entre phases de développement et d'exécution, structuration en niveaux de complexité et égale opportunité syntaxique négligées, absence de synergie entre les outils DUF proposés par la communauté scientifique IHM et les infrastructures d'exécution de la communauté intergiciel et systèmes répartis.

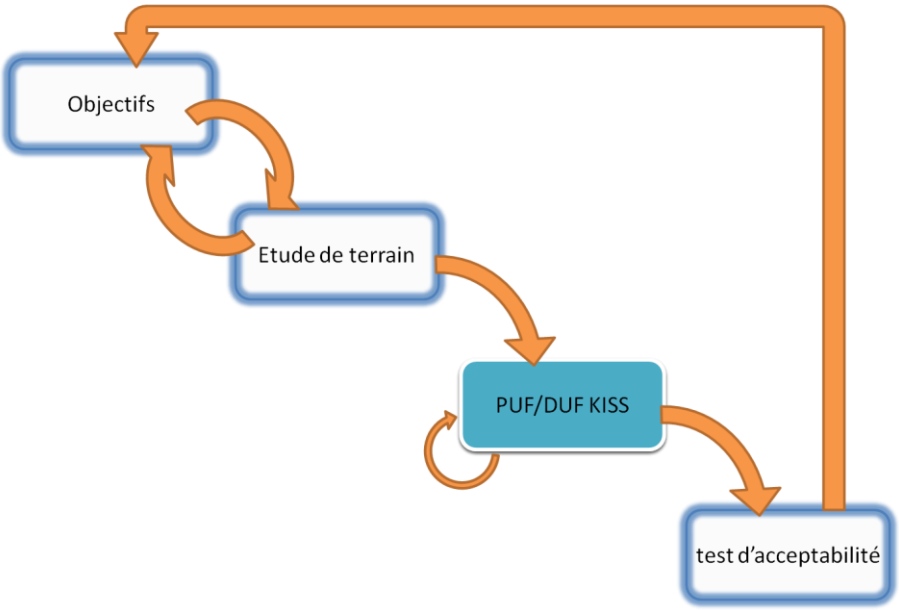


Figure IV-1. Localisation de KISS dans notre approche.

Dans ce chapitre, nous présentons KISS (Knit Your Ideas Into Smart Spaces), un outil de développement par l'utilisateur final, qui répond, non pas à toutes les lacunes relevées au chapitre précédent, mais à un sous-ensemble d'entre elles motivé par la priorité des besoins et la faisabilité dans le temps imparti à cette thèse. En particulier, nous avons mis l'accent sur l'aide à la mise au point, à l'égale opportunité syntaxique et à la coopération avec une infrastructure d'exécution issue de la communauté scientifique intergiciel et systèmes répartis, CONTINUUM. Les aspects GLUF et le support à la coopération multi-utilisateur jugés non prioritaires à court terme, n'ont pas été considérés.

Ce chapitre est structuré comme suit : Dans la section I, nous décrivons CONTINUUM : ses principes, l'intergiciel WCOMP qui sert d'appui, et son architecture. La description

de l'architecture explicite les relations de KISS avec son infrastructure d'exécution. C'est aussi l'opportunité d'indiquer la présence de trois points de contrôle ouverts sur l'utilisateur final. À chaque point de contrôle, nous faisons correspondre une classe de méta-IHM : méta-IHM réflexe, méta-IHM tactique et méta-IHM stratégique. KISS est une méta-IHM tactique. La section II décrit un exemple de méta-IHM réflexe. En III, nous présentons KISS du point de vue de l'utilisateur final suivi en IV, de la description de sa mise en œuvre technique. L'évaluation de KISS est traitée au chapitre suivant.

I – CONTINUUM, l'infrastructure d'exécution

L'infrastructure d'exécution CONTINUUM traite du problème général de la continuité de service sous l'angle de deux requis clefs de l'intelligence ambiante : d'une part, l'hétérogénéité des équipements électroniques et des services numériques, d'autre part l'adaptation dynamique aux changements d'état des mondes physiques et numériques[35]. KISS est l'un des services de cette infrastructure d'exécution, mais aussi l'un des trois points d'intervention de l'utilisateur dans le processus d'adaptation assuré par CONTINUUM. L'infrastructure d'exécution CONTINUUM s'appuie sur l'intergiciel WCOMP (Hourdin et al. 2008) (Cheung 2009), (Tigli et al. 2009), (Tigli et al. 2011) dont on présente les principes avant de décrire CONTINUUM.

I-1 Principes directeurs de WCOMP

WCOMP implémente un modèle de service et de composant dont la composition et la reconfiguration dynamiques sont exprimées à l'extérieur de la logique applicative et dont les échanges s'appuient sur une communication événementielle :

- L'*approche à composant* est retenue au nom de la réutilisabilité. En effet, un composant est une unité de composition dotée d'interfaces programmatiques contractuelles, déployable de manière indépendante et susceptible d'être composée par des tiers (Szyperski 2002). On peut donc recruter ou remplacer un composant tout comme on installe ou remplace une ampoule dans le monde réel ou un équipement électronique dans un espace intelligent.
- L'*approche à service* est retenue pour la découverte et la disparition dynamiques d'entités. Un service, selon (Bieber 2001), « est un comportement défini par un contrat qui peut être implémenté et fourni par un composant quelconque pour être utilisé par un autre composant, fondé sur ce seul contrat ». Autrement dit, dans un espace intelligent, un service est assuré par le composant, choisi dynamiquement, qui peut le remplir au mieux.

[35] URL du projet CONTINUUM : http://continuum.unice.fr

- La *communication événementielle* est retenue par la nécessité de construire un système réactif au changement. En effet, l'appel de méthode usuel suppose connaître à l'avance le séquençage des traitements, ce qui est rarement le cas dans le contexte opportuniste des espaces intelligents. La communication événementielle présente un double avantage : (1) l'inversion du contrôle (mécanisme bien connu des « callbacks » des Interfaces Homme-Machine, en réponse aux comportements non déterministes des utilisateurs) et (2) découplage des relations de dépendance entre les composants, découplage qui facilite la reconfiguration.
- L'*approche AOP (Aspect Oriented Programming)* est retenue pour l'expression de la (re)composition des assemblages de composants qui, par essence, est une préoccupation transverse à la logique applicative des composants proprement dits. Cette approche présente un double avantage : d'une part, elle respecte le principe d'isolation (blackbox) des composants et de là, leur réutilisabilité ; d'autre part, l'expression de la (re)composition peut être générée non seulement dynamiquement mais aussi par plusieurs experts logiciels garantissant ainsi une adaptation fine au plus près des besoins.

Ces choix de principes étant justifiés, WCOMP les met en application en termes de composant, de container, d'aspect d'assemblage, de designer et de service composite :

- Un *composant WCOMP* possède des métadonnées (comme son type, son nom, sa localisation, son appartenance) et une interface programmatique contractuelle constituée de la liste de ses ports d'entrée et de la liste des événements qu'il produit en sortie. Les composants WCOMP sont reliés par des connecteurs qui, comme l'illustre la figure IV-2, associent des événements de sortie à des ports d'entrée. Un graphe de composants WCOMP reliés par des connecteurs forme un assemblage qu'encapsule une entité logicielle particulière de WCOMP appelée « container ».

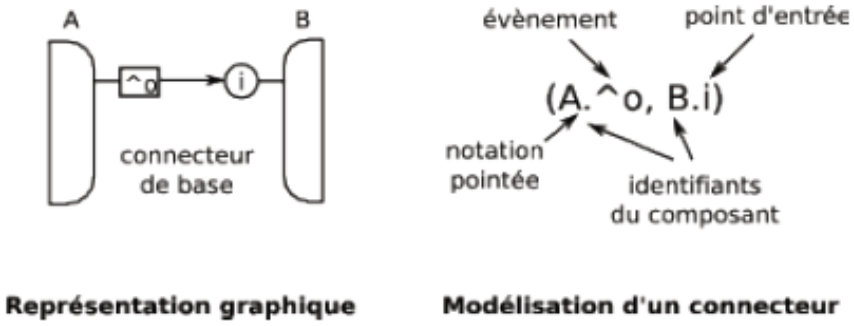


Figure IV-2. Le concept de connecteur dans WCOMP [Extrait de (Cheung 2009)].

- Un *container* permet la création, la destruction et la reconfiguration de l'assemblage qu'il encapsule : les composants de cet assemblage peuvent être chargés ou déchargés, ajoutés ou retirés, ou encore (re)connectés

dynamiquement. La (re)composition de l'assemblage d'un container est exprimée dans un langage dédié dit Aspect d'Assemblage (AA). Un AA, on l'a dit, est extérieur à la logique applicative des composants. Il est interprété par une entité logicielle particulière de WCOMP appelée *AA Designer*.

- Par analogie avec la terminologie de l'AOP, un *Aspect d'Assemblage* comprend un *point de coupe* qui spécifie les conditions de déclenchement d'une adaptation, et un *greffon* ou actions d'adaptation. Un point de coupe est un ensemble d'expressions régulières qui portent sur les identifiants des composants et sur leurs ports. Il sert de filtre de sélection des composants candidats à la reconfiguration (voir l'illustration des figures IV-3 et IV-4). Si la liste des candidats est vide, le greffon n'est pas appliqué. Le greffon exprime au moyen d'opérateurs proches d'un ADL[36], les modifications structurelles à effectuer sur l'assemblage : ajout ou retrait de composant, suppression de liaison ou ajout de liaison entre événement de sortie et port d'entrée.

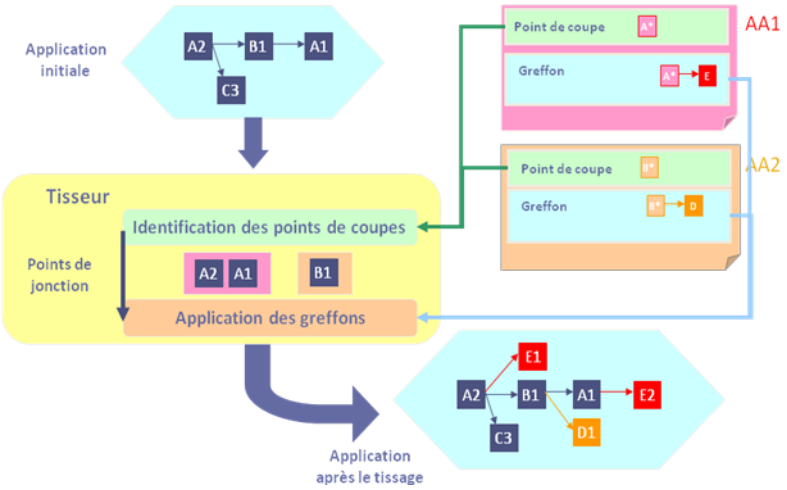


Figure IV-3. Principe du mécanisme de reconfiguration de WCOMP réalisée par l'AA Designer. En haut à droite, deux Aspects d'Assemblage interprétés par l'AA Designer de nom respectif AA1 et AA2. Le point de coupe A* de AA1 désigne tous les composants du container applicatif initial (en haut à gauche de la figure) dont l'identifiant commence par A. S'il existe de tels composants, alors le greffon de AA1 spécifie que ces composants (en l'occurrence, A1 et A2) doivent être reliés à une instance de composant E. E n'étant pas présent dans le container initial, il est créé, chargé, instancié et relié dans le container applicatif par le AA Designer pour produire *in fine* une nouvelle configuration (en bas à droite).

[36] Architecture Description Language.

```
composant1 = *?type=lampe

composant2 = *?type=interrupteur

advice interrupteur_lampe(composant1, composant2) :

composant2.^StateChange -> (composant1.setState)
```

Figure IV-4. Un aspect d'assemblage exprimant la création d'une liaison entre tous les interrupteurs et toutes les lampes du domicile. La variable *composant1* désigne tous les composants de type *lampe* et la variable *composant2* tous les composants de type *interrupteur*. S'il existe des interrupteurs et des lampes, alors appliquer l'advice (greffon) suivant de nom *interrupteur_lampe* comportant deux paramètres *composant1* et *composant2* : pour tout exemplaire d'interrupteur de *composant2*, relier son événement de sortie *StateChange* au port d'entrée *setState* de tout exemplaire de lampe de *composant1*. Autrement dit : sur événement *StateChange* produit par un interrupteur, appeler la méthode *setState* d'une lampe.

- Par souci d'efficacité, les *composants d'un container partagent le même espace d'adressage et s'exécutent au sein d'un même processus*. Pour cette raison, ils sont dits « composants locaux légers ». Or un espace intelligent, qui est constitué d'une multitude de dispositifs computationnels, n'est pas un système centralisé. En réponse au requis de distribution (et d'hétérogénéité), les dispositifs de l'espace intelligent sont supposés répondre au *protocole UPnP*. Ils sont dès lors représentés dans les assemblages par des composants particuliers appelés *composants proxy*.
- Une entité logicielle particulière de WCOMP, appelée *UPnP Designer*, a pour fonction de générer, charger et instancier dynamiquement un *composant proxy* chaque fois qu'il reçoit, via les mécanismes d'UPnP, une notification de l'apparition d'un dispositif UPnP dans l'espace intelligent. Inversement, quand un service UPnP disparaît, l'UPnP Designer supprime du container, le composant proxy correspondant. Dès lors, tout composant proxy est traité comme tout composant de container. En particulier, il peut être référencé dans les AAs de reconfiguration comme dans l'exemple de la Figure IV-4, sachant que les lampes et les interrupteurs du monde physique sont des dispositifs UPnP. La Figure IV-5 en illustre le principe.

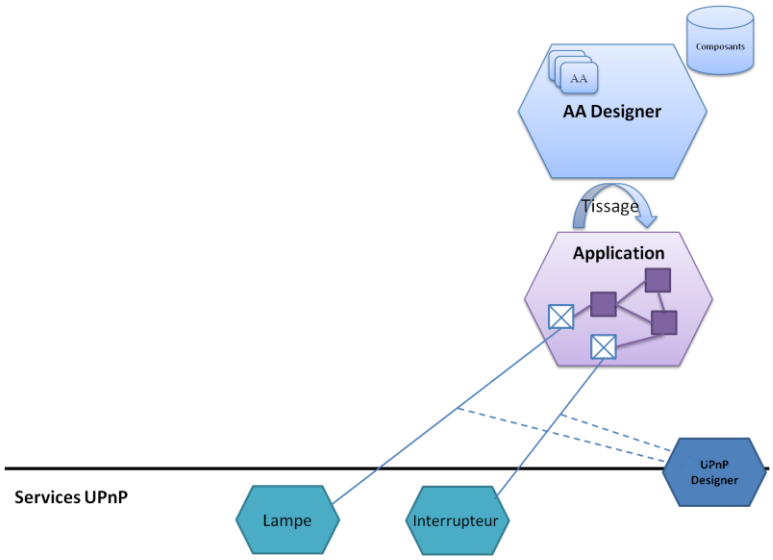


Figure IV-5. Les dispositifs de l'espace intelligent sont compatibles UPnP. L'*UPnP Designer* de WCOMP a la charge de les représenter dans un container applicatif sous forme de *composants proxy* (dénotés dans ce schéma par un carré avec ses diagonales).

- Un container peut à son tour être exporté comme un service appelé *service composite*. Comme le montre la figure IV-6, un service composite exporte une interface fonctionnelle pour communiquer avec les fonctions offertes par le service et une interface de contrôle (dite structurelle) pour accéder et éventuellement modifier la structure de l'assemblage de composants qu'il encapsule. Le concept de service composite permet de construire de nouvelles abstractions réutilisables et réparties. La figure IV-7 montre le principe général d'un espace intelligent construit au-dessus de WCOMP. L'infrastructure d'exécution CONTINUUM en est une instanciation particulière.

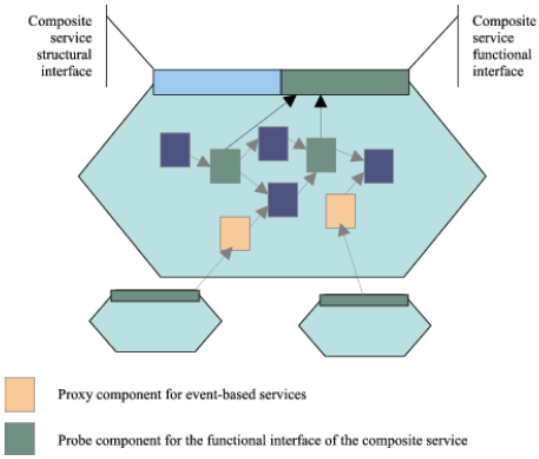


Figure IV-6. Un container WCOMP encapsulé dans un service composite WCOMP. On notera la présence des *composants proxy* représentant des services (dispositifs) UPnP.[Tiré de (Tigli et al. 2011)]

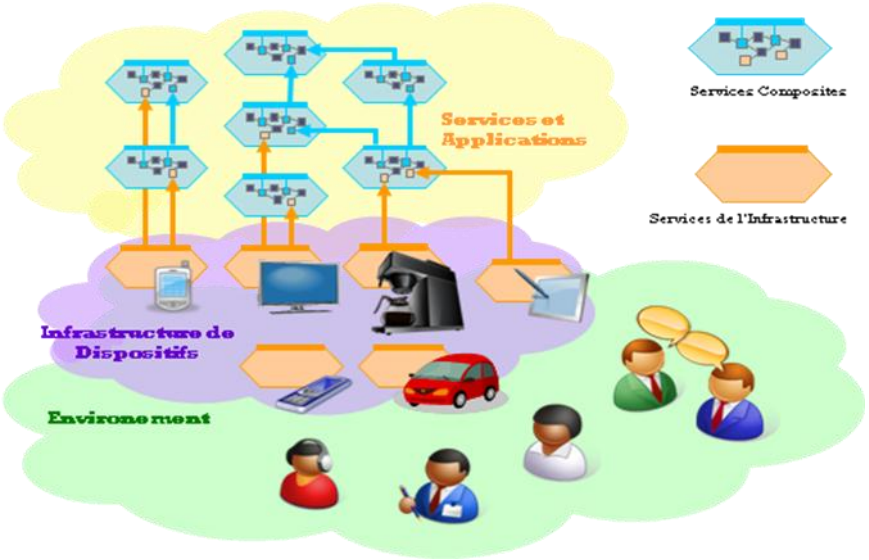


Figure IV-7. Un espace intelligent construit au-dessus de l'intergiciel WCOMP. [Tiré de (Tigli et al. 2009)]

I-2 Architecture logicielle de CONTINUUM

I-2-1 Présentation générale

L'infrastructure d'exécution CONTINUUM est un ensemble de services qui tirent parti des mécanismes d'adaptation dynamique de WCOMP pour assurer la continuité de service dans les espaces intelligents. Au chapitre 1, rappelons-le, nous avons milité pour un équilibre maîtrisé entre totale autonomie du système et intervention humaine systématique dans le processus d'adaptation. La couverture fonctionnelle des services de CONTINUUM et leurs relations tiennent compte de ce requis. La figure IV-8 en donne une représentation architecturale globale :

- Les *services UPnP* de l'espace intelligent (en bas de la figure) encapsulent les dispositifs (équipements/objets) communicants de l'espace (lampes, interrupteurs, réveils, etc.).
- L'*application* est un service composite WCOMP qui encapsule l'assemblage des composants applicatifs en cours d'exécution.
- Deux designers remplissent les fonctions présentées ci-dessus en I-1 : l'*AA Designer* calcule et effectue les (re)configurations de l'assemblage de l'application ; l'*UPnP Designer* fait le pont entre la notion de composant WCOMP et les services UPnP de l'environnement physique sous forme de *composants proxy*.
- En haut à droite, une base des composants disponibles sur étagère regroupe les composants susceptibles d'être recrutés dynamiquement par l'AA Designer. L'hypothèse implicite est que les métadonnées de ces composants

sont exprimées dans une ontologie unique : l'alignement sémantique n'est pas supporté dans la version actuelle de CONTINUUM.

- À ces services de base, viennent se greffer trois unité fonctionnelles que nous allons décrire ci-dessous. Il s'agit de la Base de Connaissances (BdC), du Gestionnaire de Contexte (GC) et de trois méta-IHM, points de contrôle ouverts sur l'utilisateur final : méta-IHM-tactique-GC, méta-IHM-tactique-BdC et méta-IHM réflexe.

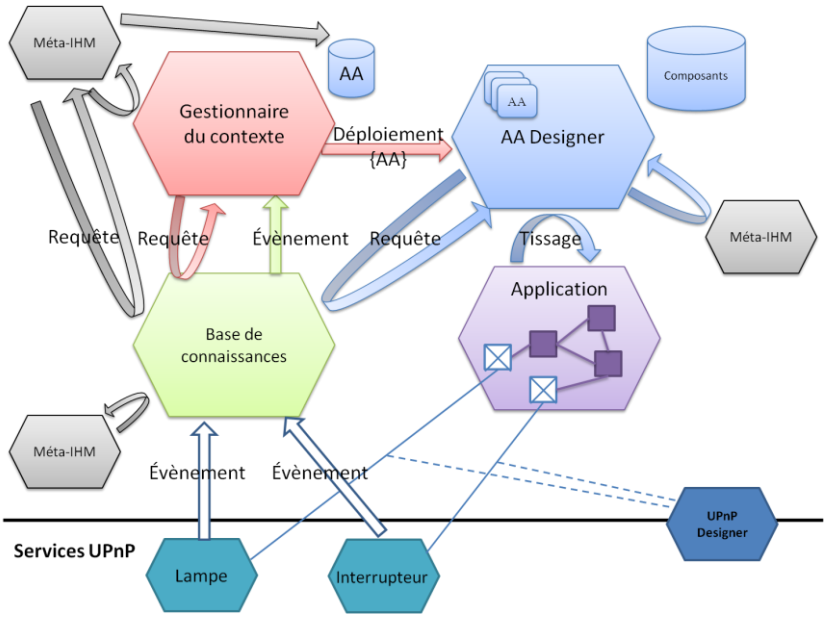


Figure IV-8. Représentation de l'architecture de CONTINUUM. Les hexagones représentent des services, les flèches des échanges entre services.

I-2-2 La Base de Connaissances CONTINUUM

La Base de Connaissances CONTINUUM (BdC) maintient un modèle déclaratif de l'espace intelligent. Elle s'appuie pour cela sur les langages RDFS et SPARQL, deux standards du W3C, le premier pour déclarer des faits, le second pour interroger la BdC sur l'existence de faits. Ce choix technique offre l'avantage d'exploiter les capacités de raisonnement et d'inférences du Web sémantique. Les événements de modification de la base de faits (par exemple, faisant suite à l'arrivée ou au départ d'un dispositif UPnP, lampe ou réveil), de même les requêtes, typiquement « existe-t-il un dispositif d'affichage ? », se font via le protocole UPnP.

La BdC CONTINUUM représente l'état de l'espace intelligent sous forme de faits RDFS. Un fait RDFS est un triplet comprenant un sujet, un prédicat et un objet, tous trois relevant d'une même ontologie. Une ontologie est une description formelle d'un domaine d'intérêt à partir d'un ensemble fini de classes et de relations entre ces classes qui représentent les entités pertinentes de ce domaine ainsi que leurs propriétés. En l'état, l'ontologie noyau utilisée dans CONTINUUM comprend quatre classes de base : Personne, Dispositif, Service et Tâche (Benyelloul et al. 2010). Chacune de ces classes comprend un ensemble de propriétés inspiré du modèle 5W1H

(what, when, where, who, why, how) (Jang et al. 2005)[37]. La figure IV-9 est une représentation graphique de l'ontologie noyau de la BdC CONTINUUM. Comme le montre la Figure IV-10, ce noyau peut être facilement enrichi par spécialisation des classes de base.

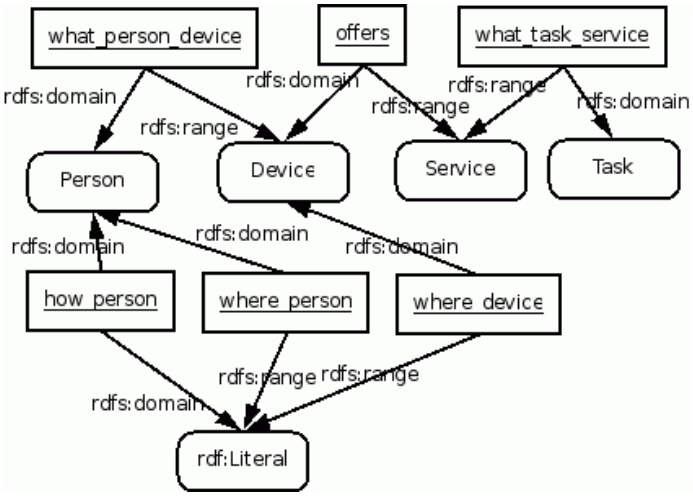


Figure IV-9. Ontologie noyau de la BdC CONTINUUM. La classe *Person* dénote l'ensemble des utilisateurs présents dans l'espace intelligent. Ils sont localisés (propriété *where-person*), ils utilisent des dispositifs (propriété *what-person-device*) et ont un comportement physique (propriété *how-person*). La classe *Device* représente les dispositifs physiques disponibles. Ils offrent (propriété *offers*) des services et sont localisés (*where-device*). La classe *Service* représente les services logiciels offerts par des dispositifs ou requis pour la réalisation d'une tâche. La classe *Task* représente les objectifs à atteindre et requiert un ensemble de services pour les réaliser (propriété *what-task-service*).

[37] Alors que ce modèle fait figure de quasi nouveauté dans la littérature sur la modélisation du contexte, la communauté scientifique ne fait que redécouvrir ces questions : Quintilien, rhéteur et pédagogue latin du 1er siècle après J.-C., a largement disserté sur ces éléments de rhétorique, éléments que Socrate, cinq siècles plus tôt, plaçait déjà au cœur du raisonnement critique.

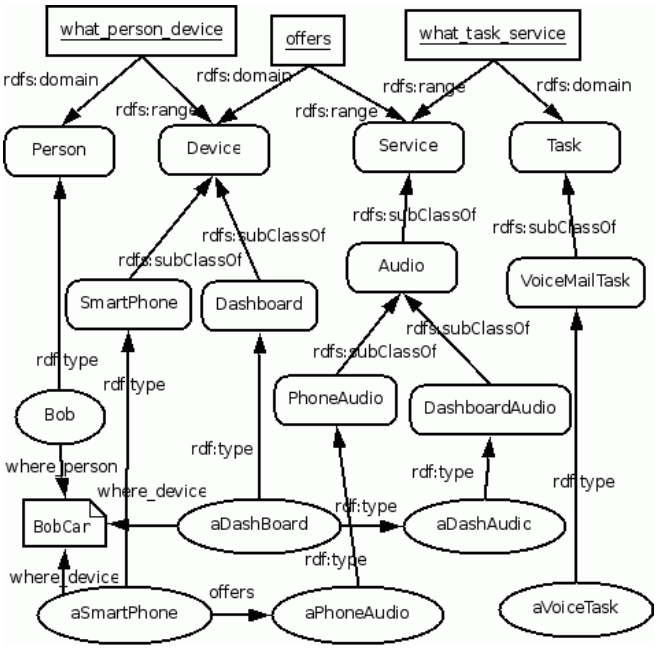


Figure IV-10. Exemple d'ontologie pour la modélisation d'un véhicule intelligent où les classes et sous-classes sont représentées par des rectangles et les instances par des ovales.

La base de faits est interrogée sous forme déclarative au moyen de requêtes SPARQL, conjonction de prédicats de la logique du premier ordre. Par exemple, la requête "quels dispositifs offrent un service de type audio dans la voiture de Bob ?", s'exprime ainsi en SPARQL :

SELECT ?d WHERE { ?d type Device .

?d where_device BobCar .

?d offers ?s .

?s type Audio . }

L'évaluation de cette requête par le moteur SPARQL retourne, pour l'exemple de la figure IV-10, le résultat {aSmarthPhone, aDashBoard}. Notons qu'une requête SQL classique aurait fourni un résultat vide car, sans inférence, aucun fait connu ne répond directement à la question. En l'occurrence, le moteur SPARQL, qui applique un algorithme de saturation, reconnaît que :

$SmarPhone \subseteq Device \quad DashBoard \subseteq Device$

$PhoneAudio \subseteq Audio \quad DashBoardAudio \subseteq Audio$

À partir de faits existants et de règles de déduction, il infère de nouveaux faits (d'où l'expression « algorithme de saturation ») et se trouve en mesure de produire le résultat :

{aSmarthPhone, aDashBoard}

Comme le montre l'architecture de CONTINUUM (figure IV-8), la BdC reçoit des événements UPnP en provenance des dispositifs UPnP de l'espace intelligent. Le traitement de ces événements a deux effets : (1) modification éventuelle de la base de faits, (2) envoi au Gestionnaire de Contexte (GC) de deux types d'événements : événements de haut niveau correspondant aux modifications de la base de faits, et événements originaux des dispositifs UPnP si le GC en a exprimé l'intérêt. Enfin, la BdC répond aux requêtes issues notamment du Gestionnaire de Contexte (mais aussi de KISS, comme nous le verrons plus loin).

I-2-3 Le Gestionnaire de contexte de CONTINUUM

Le Gestionnaire de Contexte (GC) s'appuie sur les concepts introduits par (Rey 2005) et (Coutaz et al. 2005) : un contexte est un graphe de situations qui partagent les mêmes ensembles de prédicats et d'entités. Dans CONTINUUM, ces entités sont les classes et instances de classes modélisées dans la BdC (Personne, Dispositif, etc.) et les prédicats sont des prédicats exprimables dans les termes de la BdC. Au sein d'un contexte donné, il y a changement de situation sur changement de valeur de prédicat et/ou sur changement de la cardinalité de l'ensemble des entités, par exemple l'arrivée d'un nouveau dispositif dans l'espace intelligent. Il y a changement de contexte si la nature de ces ensembles change (ce qui correspond, *grosso modo*, à un changement d'ontologie).

Le GC a pour mission de détecter les changements de situation, d'identifier la nouvelle situation et de là, s'il en est besoin, de produire un plan d'adaptation correspondant à la nouvelle situation et de le faire exécuter. Pour cela, le GC interroge la BdC (requête SPARQL) sur réception de notification de changement d'état en provenance de la BdC notamment. À chaque situation correspondent des AAs mémorisés dans une base d'AAs (en haut à gauche de la figure IV-8). Sur identification d'une nouvelle situation, le GC retire de l'AA Designer les AAs de l'ancienne situation, puis déploie sur l'AA Designer les AAs correspondant à la nouvelle. L'AA Designer, sollicité par ce nouveau déploiement, interprète les AAs déployés avec, comme conséquence, la (re)configuration du service composite Application.

La base des plans d'adaptation « situation- AAs » n'est pas fixe. Elle évolue selon deux procédés complémentaires : d'une part, par un service d'apprentissage des comportements humains (non implémenté dans la version actuelle de l'infrastructure), d'autre part, par DUF. C'est ici que KISS intervient en tant que composante de méta-IHM.

I-2-4 Méta-IHM comme points de contrôle humain dans les mécanismes d'adaptation de CONTINUUM

> Une « méta-IHM regroupe l'ensemble des fonctions (avec leur interface utilisateur) nécessaires et suffisantes pour évaluer et contrôler non seulement l'état mais aussi le comportement de systèmes ambiants et notamment leur processus d'adaptation. » (Coutaz 2007).

Cet ensemble est méta- parce qu'il sert d'arche à tout l'espace intelligent. Il donne à l'utilisateur final les poignées pour intervenir aussi bien sur le comportement fonctionnel de l'espace intelligent que sur son comportement interactionnel. En somme, une méta-IHM est aux espaces intelligents, ce que le desktop ou le shell est aux systèmes centralisés usuels.

Dans CONTINUUM, la *méta-IHM intervient comme point de contrôle humain* à chaque niveau d'adaptation que permet l'infrastructure. De manière originale et par analogie avec le modèle des comportements cognitifs de Rasmussen (Rasmussen 1986), l'infrastructure d'exécution offre trois niveaux d'adaptation.

1. Le premier niveau (*skill level*) correspond aux comportements réflexes selon lesquels les actions sont réalisées de manière inconsciente en réaction immédiate aux phénomènes perçus.
2. Le second niveau (*rule-based knowledge*) désigne les comportements tactiques qui s'appuient sur des ensembles de règles apprises par l'expérience. À ce niveau, il s'agit, pour une situation connue, de choisir le plan à appliquer parmi l'ensemble des plans solutions.
3. Le troisième niveau (*deep knowledge*) correspond aux comportements stratégiques : la situation est inconnue. Il convient dès lors d'élaborer un nouveau plan d'actions en faisant appel à la connaissance profonde.

À l'évidence, ces niveaux traduisent des efforts cognitifs croissants auxquels correspondent des temps de réponse croissants. L'infrastructure CONTINUUM reproduit ces mêmes principes (Ferry et al. 2011):

1. L'*adaptation réflexe* est assurée par l'AA Designer qui réinterprète les AA déployés sur apparition/disparition d'un composant proxy par l'UPnP Designer. Du point de vue de l'utilisateur final, la latence de l'adaptation réflexe doit être quasi-imperceptible (c.-à-d. inférieure à 80ms). À ce niveau réflexe, correspond une *méta-IHM réflexe* dont on verra un exemple dans la section suivante.
2. L'*adaptation tactique* est assurée par le GC et la BdC qui, ensemble, permettent de détecter des changements de situations, d'identifier la nouvelle situation parmi celles qui sont connues, et de choisir dans la base des scriptes d'AAs, celui qui convient. Du point de vue de l'utilisateur final, la latence de l'adaptation tactique est supposée être de l'ordre de la seconde. Nous convenons d'introduire *une méta-IHM-tactique* qui se décompose en deux

parties : la *méta-IHM-tactique-GC* qui doit permettre à l'utilisateur final de programmer des scriptes d'AAs pour des situations données (ce sera le rôle de KISS) et la *méta-IHM-tactique-BdC* pour visualiser et modifier la BdC (Benyelloul, Jouanot, and Rousset 2010). La figure IV-11 en donne une illustration.

3. L'*adaptation stratégique* est assurée par un service d'apprentissage non implémenté dans la version actuelle. Une *méta-IHM-stratégique* servirait alors à guider le service d'apprentissage en cas de doute et plus généralement à vérifier et corriger les faits appris. Dans cette recherche doctorale, nous n'avons pas travaillé sur cette dimension.

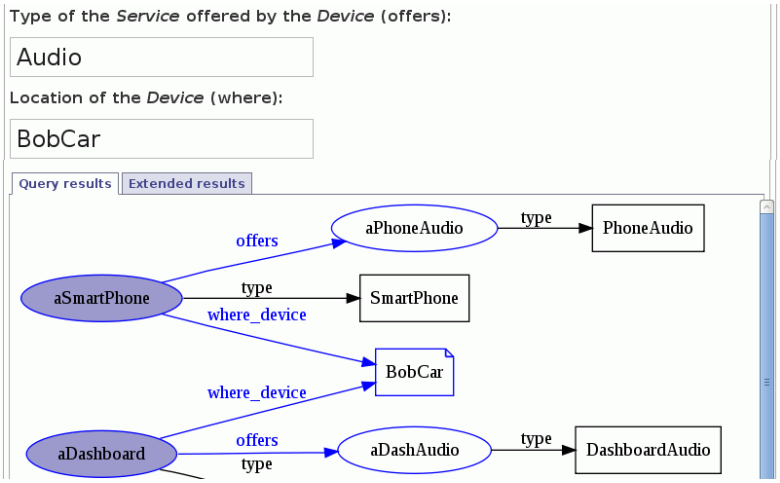


Figure IV-11. Méta-IHM-tactique-BdC destinée à la mise au point de la BdC plutôt qu'à l'utilisateur final d'espace intelligent [tiré de (Benyelloul, Jouanot, and Rousset 2010)].

En résumé, l'infrastructure d'exécution CONTINUUM introduit trois niveaux d'adaptation (réflexe, tactique et stratégique) par analogie avec les niveaux de compétences humaines avec, en regard, des exigences de latence spécifiques (Ferry 2011), (Ferry et al. 2010). À chacun de ces niveaux, nous faisons correspondre un point de contrôle humain, par analogie aux points de contrôle système qui permettent des actions correctives, en l'occurrence par l'utilisateur. À chaque point de contrôle, nous définissons une composante de méta-IHM : méta-IHM réflexe, méta-IHM tactique et méta-IHM stratégique. Dans ce qui suit, nous présentons un exemple de méta-IHM réflexe suivi d'une description approfondie de la méta-IHM tactique-GC KISS.

II – Un exemple de méta-IHM réflexe

La méta-IHM réflexe que nous avons réalisée permet de contrôler le lancement et l'arrêt d'une application de tri de photos, Photo-Browser, de même la répartition de son IHM en fonction des dispositifs d'interaction disponibles dans l'espace intelligent (Balme 2008). Cette application correspond à la séquence « vacances au restaurant »

de notre scénario de référence (voir détail en Annexe II-3). Nous décrivons Photo-Browser du point de vue de l'utilisateur, suivi de sa mise en œuvre au-dessus de l'infrastructure CONTINUUM.

II-1 L'interaction dans Photo-Browser

La figure IV-12 illustre une séquence d'interaction avec Photo-Browser. Le démarrage de l'application s'effectue en posant un SmartPhone sur la table MERL interactive multipoint. *Ce contact est détecté et traité par la méta-IHM réflexe de l'espace intelligent*. Les photos s'affichent sur la table. L'utilisateur peut alors déplacer les photos, de même les réorienter tout en changeant leur taille. l'IHM de Photo-Browser, qui utilise une seule surface de rendu, est alors centralisée.

L'utilisateur peut ordonner la répartition de l'IHM de Photo-Browser entre la table et le mur (géré par un PC) par un geste de balayage de la main allant de la table vers le mur. *Ce geste, capturé par une Kinect*[38]*, est reconnu par la méta-IHM réflexe de l'espace intelligent* : la photo qui était sélectionnée sur la table se trouve maintenant affichée à la fois sur le mur et la table. Toute nouvelle sélection de photo sur la table est synchronisée avec le mur. L'IHM de Photo-Browser est maintenant répartie sur deux ressources d'interaction de l'espace intelligent : la table et le mur.

L'utilisateur peut introduire son Smartphone dans l'espace intelligent en le posant sur la surface de la table. *Ce geste, reconnu par la méta-IHM reflexe de l'espace intelligent*, permet d'utiliser le SmartPhone comme télécommande : l'utilisateur navigue séquentiellement dans l'espace des photos grâce aux boutons « next » et « previous » affichés sur l'écran du SmartPhone : les photos sont affichées de manière synchronisée à la fois sur le mur et la table. L'IHM de Photo-Browser est maintenant répartie entre la table, le mur et le SmartPhone.

La *méta-IHM réflexe de l'espace intelligent permet de supprimer la table* de l'espace intelligent par un balayage de la main sur sa surface. L'IHM de Photo-Browser est maintenant répartie entre le mur et le SmartPhone.

[38] Microsoft kinect : http://www.xbox.com/fr-FR/Kinect

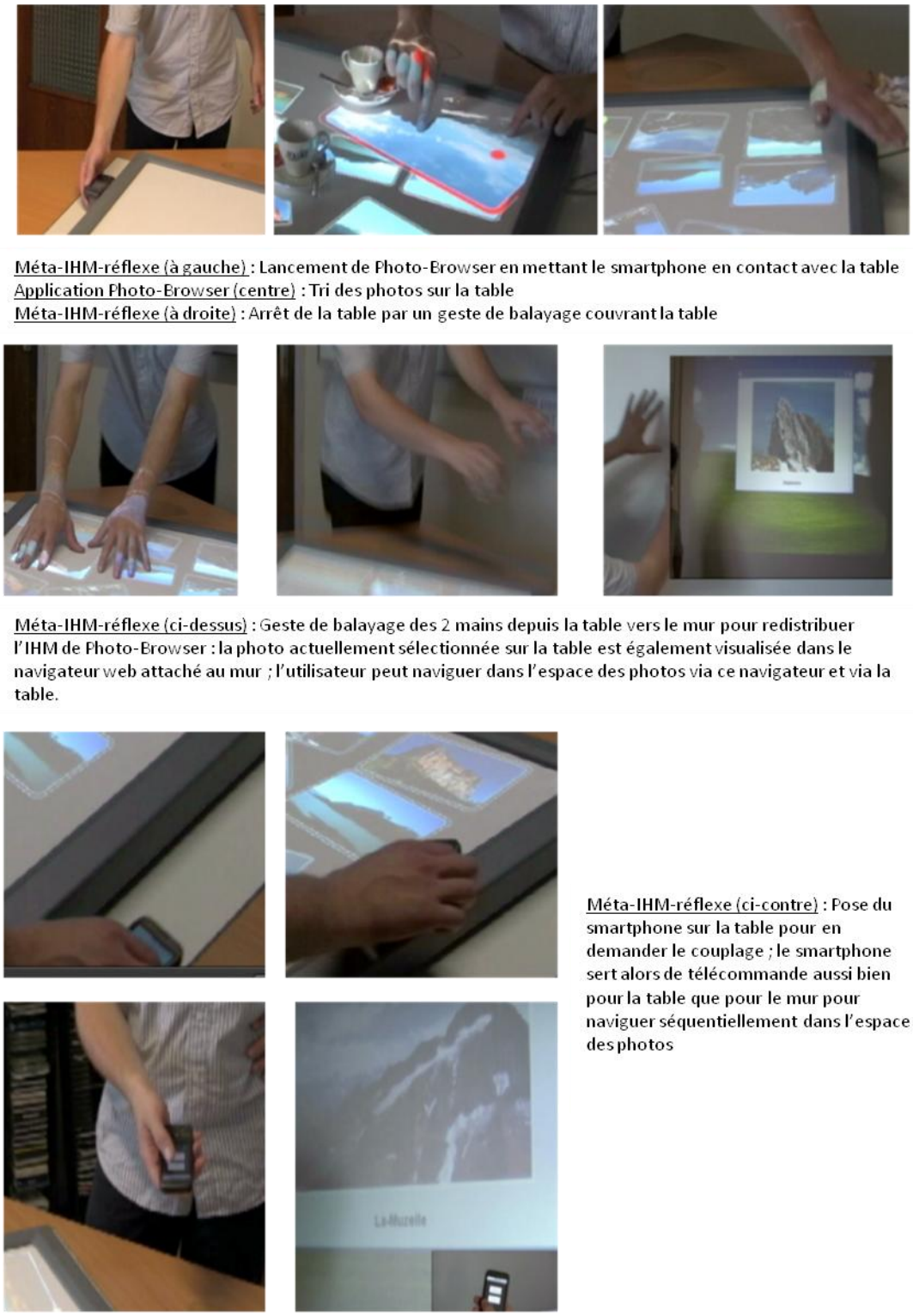


Figure IV-12. Méta-IHM-réflexe dans Photo-Browser.

II-2 Mise en œuvre

Photo-Browser est une application patrimoniale organisée selon le modèle d'architecture Seeheim (Pfaff 1985). Elle comprend les composants suivants :

- Le composant *Table*, implémenté en Tcl/Tk, assure l'interaction sur la table interactive.
- Le composant *Télécommande*, implémenté en java, correspond au SmartPhone utilisé en mode télécommande.
- Le composant *Renderer*, implémenté en HTML, affiche les photos sur le mur.
- Le composant *Contrôleur de Dialogue* (CD), conformément au modèle architectural Seeheim, assure l'enchaînement des tâches et la cohérence entre les interactions assurées par les autres composants.

La figure IV-13 montre la décomposition fonctionnelle de Photo-Browser et son lien avec la Méta-IHM réflexe. Pour être « compatible-CONTINUUM », chaque composant de Photo-Browser est encapsulé en un dispositif UPnP. Ainsi, l'application Photo-Browser est un service composite WCOMP structuré en quatre composants proxy : un proxy Table, un proxy Télécommande, etc.

Le rôle de la méta-IHM réflexe est d'interpréter, sous forme d'AAs, les gestes acquis par la Kinect (qui est un device UPnP), et de les déployer à la volée dans l'AA Designer. On obtient ainsi la reconfiguration dynamique de l'IHM de PhotoBrowser. Par exemple, un geste de balayage des deux mains depuis la table vers le mur aura pour effet de déployer un AA qui orchestre la relation entre le Renderer et le CD.

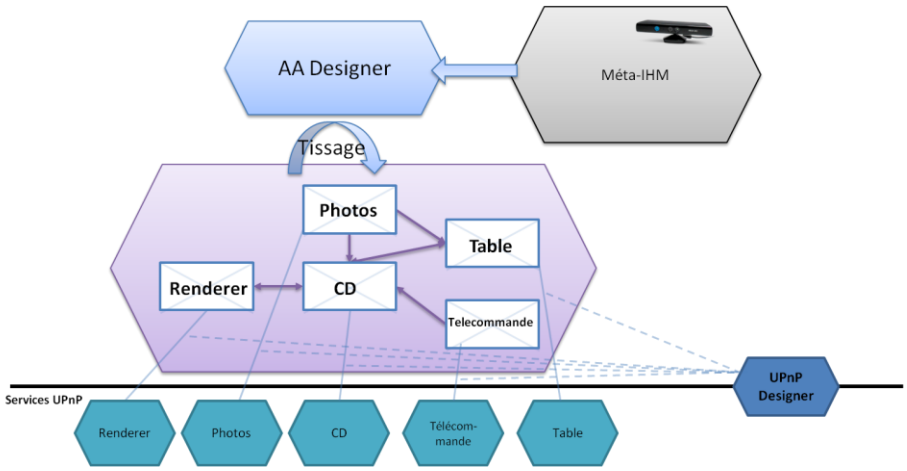


Figure IV-13. Photo-Browser dans CONTINUUM et sa relation avec sa Méta-IHM réflexe.

L'exemple de Photo-Browser est intéressant à deux égards. Il montre comment :

1. Une application patrimoniale, pour peu qu'elle soit décomposée en unités fonctionnelles bien distinctes, peut être portée sur l'infrastructure CONTINUUM ;
2. Une méta-IHM, distincte d'un code applicatif comme Photo-Browser, peut significativement augmenter la capacité de contrôle de l'utilisateur et de là, la richesse interactionnelle de cette application (ici, par la reconnaissance de gestes non prévue dans l'application d'origine).

Ayant présenté un exemple de méta-IHM réflexe, nous abordons maintenant le cas de KISS, un DUF qui joue le rôle de méta-IHM-tactique dans le processus d'adaptation de

l'infrastructure d'exécution CONTINUUM. Nous présentons KISS en deux volets : point de vue de l'utilisateur final (section III) et mise en œuvre technique (section IV).

III – KISS vu de l'utilisateur final

III-1. KISS dans l'espace problème PUF-DUF-GLUF

Dans notre espace problème présenté au chapitre précédent, KISS (Knit your Ideas into Smart Spaces) est un outil DUF qui se caractérise comme suit : il s'adresse à un utilisateur final constructeur. Sa couverture fonctionnelle inclut la programmation, la mise au point et dans une moindre mesure, la réutilisation. La mise au point peut se pratiquer au choix, dans le monde physique, dans un monde dual numérique ou dans les deux simultanément[39]. KISS ne fournit aucun support au co-développement, jugé pour l'instant, peu prioritaire. Par contre, il fait partie des rares exemples où phases de conception et d'exécution peuvent être perçues comme confondues.

Concernant la programmation, nous avons opté pour les choix suivants :

- Une *adaptabilité* de l'espace intelligent *par programmation*, soit *ex nihilo*, soit *à partir d'exemples réutilisables*. Il s'agit de *codage par spécification* avec *génération classique* mais, ce qui est plus original, le code généré est des plans d'adaptation AAs (cf. section I-2). Ces plans, maintenus dans la base d'AAs de l'infrastructure d'exécution, bien qu'*accessibles à l'humain*, sont compréhensibles pour le développeur professionnel spécialiste de WCOMP, mais ne sont pas destinés à l'utilisateur final.
- Un langage de programmation *déclaratif orienté règles* pour sa simplicité d'accès ainsi que l'ont démontrée des études sérieuses d'utilisabilité pour CAMP (Truong et al. 2004) et a Cappella (Dey et al. 2004).
- Un *langage spécifique* dont la syntaxe et le lexique sont établis dynamiquement à partir de la BdC assurant *de facto* l'*extensibilité* et une *totale correspondance* avec le lieu de vie de l'utilisateur final.
- Une technique d'interaction WIMP fondée sur l'utilisation de menus qui assure la *découverte progressive* du langage, impliquant un faible coût d'entrée puisque les unités du langage sont directement observables. Les options des menus, calculées dynamiquement à partir de la BdC, assurent du même coup un *support à la correction* : grâce aux menus fondés sur la BdC, qui est le reflet permanent de l'état de l'espace intelligent, l'utilisateur ne peut que construire des phrases sémantiquement correctes.
- Une *approche multisyntaxe avec égale opportunité* incluant une syntaxe de langage pseudonaturel (langue naturelle contrainte) et une syntaxe de langage

[39] S'il est possible, par conception, d'exécuter un programme dans les deux mondes à la fois, la version actuelle de KISS ne permet l'exécution que dans un seul monde à la fois.

visuel. Le langage visuel ayant fait l'objet d'une implémentation partielle, l'égale opportunité n'est dans les faits que *partielle*.

Le choix d'un langage pseudonaturel (LPN) et la construction de phrases LPN au moyen de menus se justifient ainsi : d'après une étude portant sur la composition de services (Gabillon 2011), les utilisateurs préfèrent exprimer ces compositions en langue naturelle, en conformité avec la vie courante. On retrouve ce même usage de langage pseudonaturel dans Camp (cf. chapitre précédent). Toutefois, la langue naturelle souffre de deux limitations : elle est ambiguë exigeant des micro-dialogues de clarification et ne permet pas le « feedforward » c'est-à-dire rendre observable l'espace du possible pour le discours à venir.

En réponse à la première difficulté, TellMe (Gil et al. 2011) propose un mécanisme de négociation. L'utilisateur saisit ses instructions dans un champ textuel et le système pose des questions si ces instructions sont ambiguës. Mais TellMe ne fournit pas de feedforward. En réponse à cette seconde difficulté, nous avons repris à notre compte une solution des années 80 à base de menus contextuels qui permettent de construire des phrases correctes en LPN (pseudo, car contrainte par le vocabulaire fourni dans les menus) (Tennant et al. 1983). La figure IV-14 montre la construction d'une requête d'interrogation d'une base de données. L'utilisateur voit ce qu'il peut exprimer à partir de ce qu'il a déjà rédigé. Cette requête est, par construction, syntaxiquement et sémantiquement correcte.

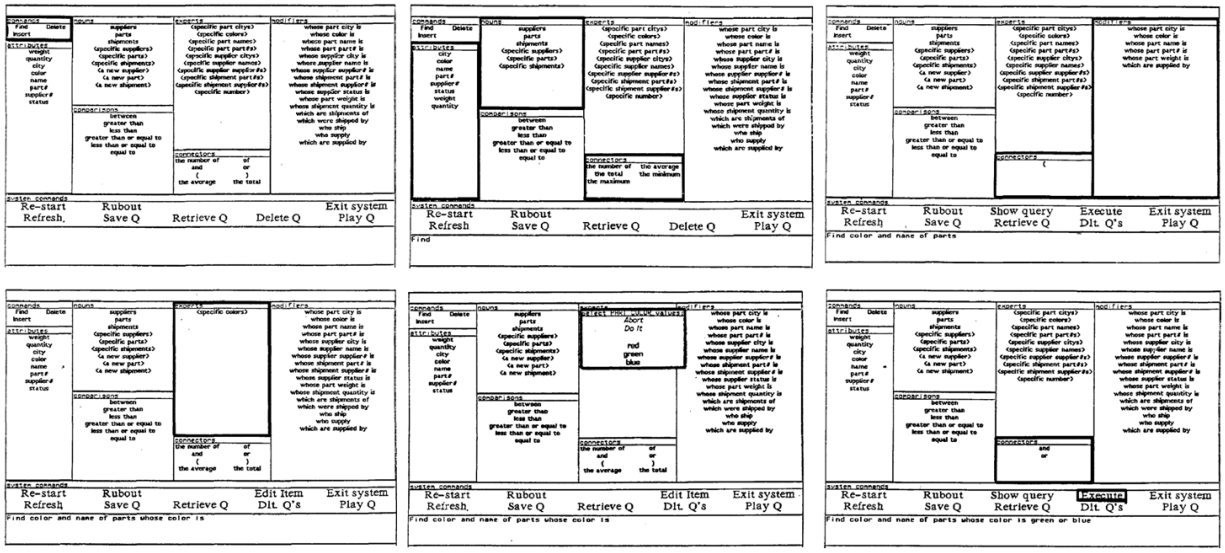


Figure IV-14. Interface proposée par Tennant pour la rédaction de requêtes auprès d'une base de données. Dans l'exemple, l'utilisateur rédige une requête par sélection des instructions dans des menus déroulants. [Tiré de (Tennant et al. 1983)].

III-2 KISS en action

La tâche centrale de l'utilisateur final consiste, rappelons-le, à programmer l'espace intelligent dans lequel il vit, et notamment à définir le comportement de son domicile

au quotidien. Du point de vue de cet utilisateur, KISS se présente donc comme un éditeur de programmes et d'un metteur au point. La figure IV-15 montre l'organisation concrète de KISS pour notre expérimentation en milieu semi-contrôlé, objet du chapitre suivant.

Figure IV-15. KISS expérimental. Sur la table de gauche, le laptop sur lequel s'exécute l'éditeur de programmes. Sur la table de droite, le laptop du metteur au point et sa tablette pour jouer sur le temps.

III-2-1 L'éditeur KISS

La figure IV-16 présente la vue générale de l'éditeur : au sommet, la phrase en cours d'édition, en bas, la liste des phrases déjà construites. En bas à gauche, un bouton pour créer une nouvelle phrase. Les figures qui suivent illustrent un scénario de construction de phrases en LPN.

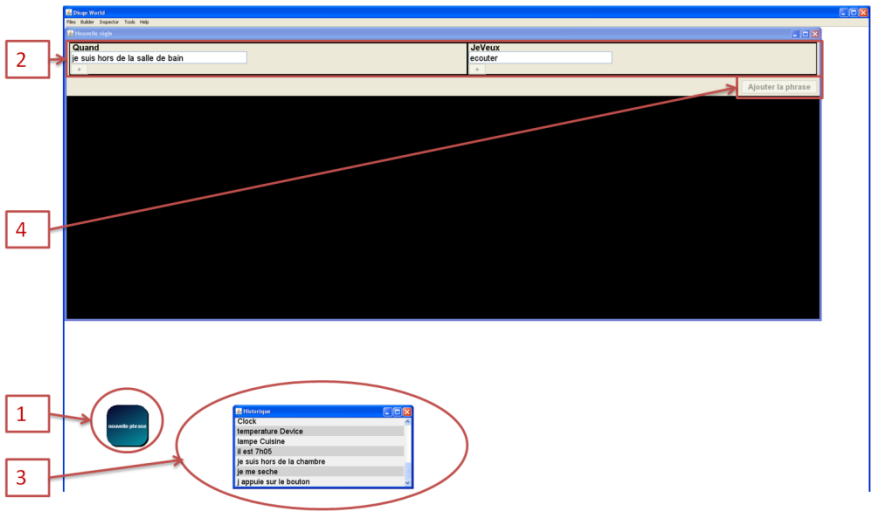


Figure IV-16. Vue d'ensemble de l'éditeur KISS. En 1, le bouton qui permet, en le cliquant, de rédiger une nouvelle phrase en langage LPN. En 2, la zone d'édition de la phrase LPN courante. En 3, la liste des phrases déjà écrites, repérées chacune par un titre. En 4, le bouton pour sauvegarder la phrase courante.

Le paradigme de programmation, nous l'avons vu, est de type déclaratif par règle. La zone d'édition comprend donc une partie condition intitulée *Quand* et une partie action, *Je veux*. Chaque partie est à son tour constituée de champs qui, une fois

remplis (au moyen de menus déroulants) peuvent faire apparaître d'autres champs. La figure IV-17 en donne une illustration.

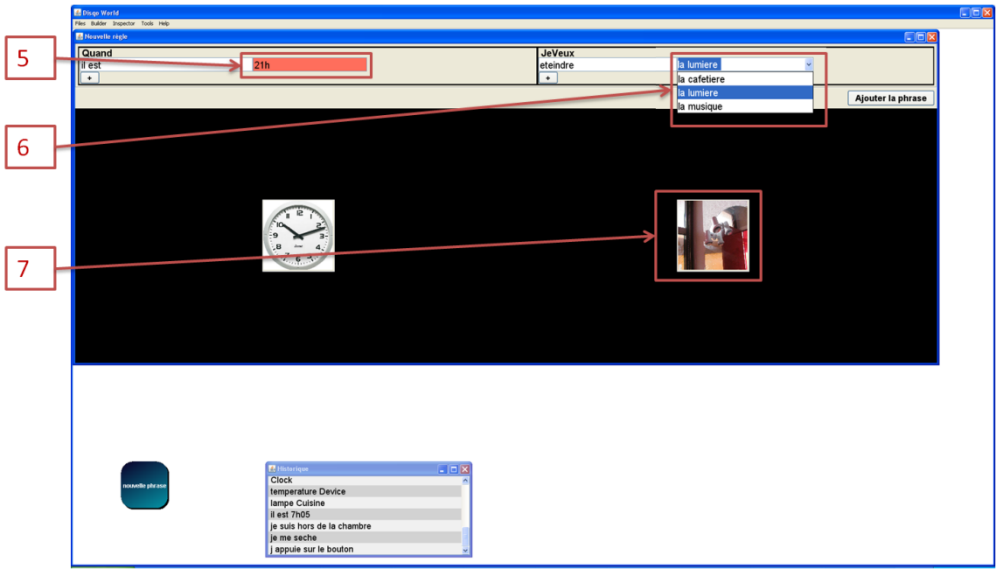


Figure IV-17. Construction d'une phrase LPN avec KISS. Pour remplir un champ donné, il suffit de placer la souris sur ce champ : un menu apparaît offrant les possibilités sémantiquement correctes parmi lesquelles l'utilisateur fait son choix. Dans l'exemple, l'élément *lumière* va être retenu comme paramètre de l'action éteindre. Le choix d'un élément de menu peut entraîner l'apparition de nouveaux arguments (champs) dont il faut spécifier la valeur. Ceux-ci sont généralement coloriés en rouge comme en 5. Lorsqu'un champ fait référence à un équipement de l'habitat, son image est affichée comme en 7 (ici, l'image de la lumière qui vient d'être choisie). La phrase actuelle est maintenant : *Quand il est 21h, je veux éteindre la lumière*.

L'utilisateur a la possibilité d'affiner les parties conditions et actions par ajout ou retrait de conditions et d'actions. La figure IV-18 montre comment. Une fois la phrase complète, il est possible de la mémoriser et de lui donner un nom. Ce nom apparaît alors dans la liste, dite *historique*, des phrases rédigées (en bas de l'écran). Toute phrase de l'historique peut être sélectionnée et éditée. KISS supporte ainsi une réutilisation d'exemples *a minima*.

La sauvegarde d'une phrase entraîne sa traduction en un plan d'adaptation « situation-AAs » où la situation vient de la partie *Quand* de la phrase et l'ensemble des AAs à déployer lorsque la situation se présentera, est issu de la partie *Je veux*. Nous reviendrons sur ce point dans la section dédiée à la description de la mise en œuvre de KISS.

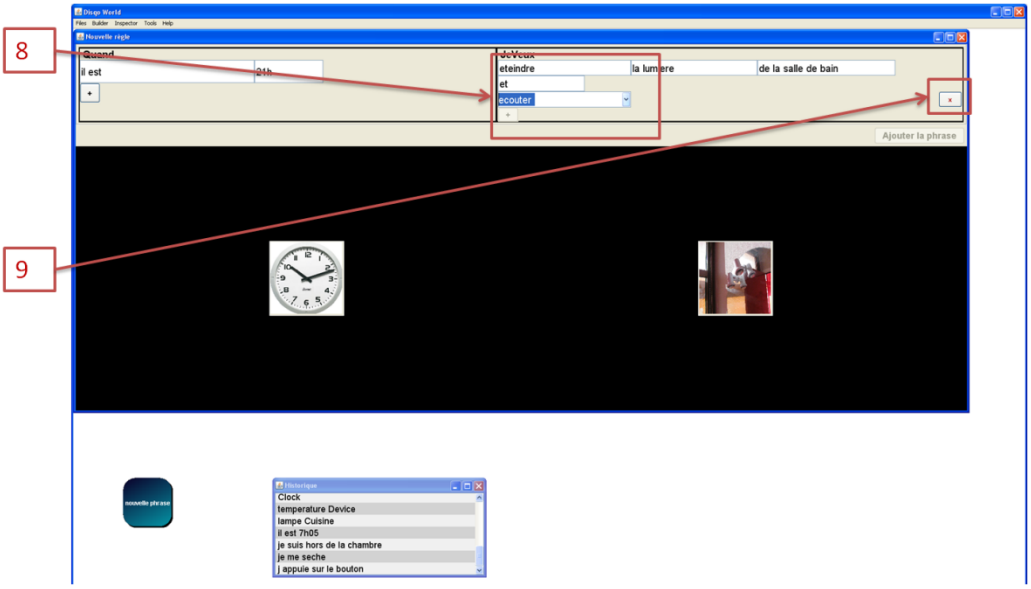


Figure IV-18. Ajout et retrait de conditions et d'actions. La sélection du bouton « + » (comme en 8) en-dessous de la portion de phrase rédigée fait apparaître un menu dans lequel il suffit de sélectionner l'opérateur logique souhaité. Les boutons « x » (comme en 9) situé en regard de chaque action et condition permettent de supprimer les actions et conditions correspondantes. La phrase actuelle est maintenant : *Quand il est 21h, je veux éteindre la lumière de la salle de bain et écouter*. Ecouter exigeant un argument, un nouveau champ va être présenté.

III-2-2 Le metteur au point KISS

Le metteur au point KISS permet d'observer l'effet d'une phrase soit dans l'environnement physique réel, soit dans une représentation duale virtuelle dit simulateur (soit dans les deux à la fois, prévu pour une version ultérieure de KISS). Le simulateur est nécessaire au test de situations qui ne correspondent pas à la situation présente de la vie réelle, par exemple l'existence d'une fuite de gaz ou un événement qui se produira plus tard dans le temps. La figure IV-19 montre le simulateur en représentation graphique 2D. Une représentation 3D, plus réaliste mais jugée moins utilisable, a également été produite (figure IV-20). La figure IV-21 montre l'IHM présentée sur la tablette graphique pour jouer sur l'heure et décider du lieu d'exécution du programme. Prenons le cas du test de la phrase « *quand il est 7h du matin et que je rentre dans la salle de bain, la lumière s'allume et la cafetière se met en marche* » et supposons qu'il est 14h00 dans le monde réel. L'utilisateur ajuste l'heure dans le champ 2 de la Figure IV-21 (afin de ne pas attendre le lendemain matin !) et choisit le mode d'exécution de la phrase (simulateur ou appartement). Ensuite, il déplace son avatar dans la salle de bain. En mode *simulateur*, le résultat de l'exécution de la phrase se constate dans la représentation graphique : la lumière de la salle de bain est maintenant allumée et la cafetière est en marche. En mode *appartement*, les actions sont appliquées à la cafetière réelle et à lumière de la salle de bain physique.

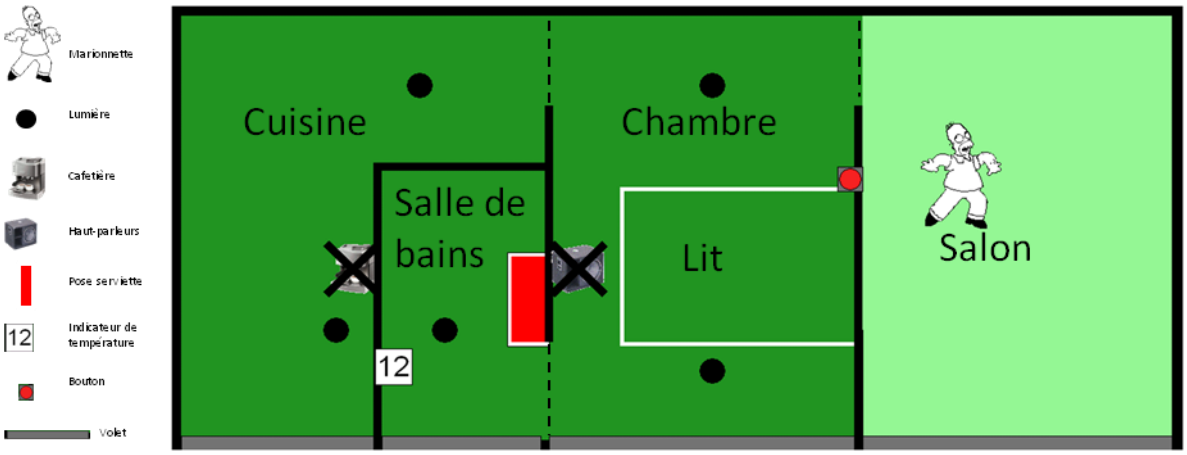


Figure IV-19. Le simulateur, représentation graphique 2D du monde réel (ici l'appartement utilisé pour notre expérimentation). On y voit les différentes pièces de l'appartement, l'état du mobilier instrumenté et la localisation de l'utilisateur : lit (augmenté de capteur de présence), volets (ouverts, semi-ouverts ou fermés), lumières (allumées ou éteintes), haut-parleurs (diffusant ou non de la musique), cafetière (en fonction ou éteinte), indicateur de température dans la salle de bain. L'utilisateur est représenté par la marionnette Homer déplaçable par glisser-déposer pour reproduire, sans avoir à se déplacer dans le monde réel, des conditions de déplacement ou de localisation comme *quand je rentre dans la salle de bain…* ou encore *quand je suis dans mon lit…*

Figure IV-20. Le simulateur en représentation graphique 3D selon une vue à la 3ème personne.

Figure IV-21. Spécification de l'heure et du lieu de test du metteur au point KISS. En 2, un champ permet de spécifier l'heure actuelle (pour se placer dans l'avenir par exemple). En 3, la sélection des flèches permet d'agir sur le coefficient d'accélération du temps (valeur courante affichée en 1) afin que l'utilisateur en situation de test n'ait pas à subir la durée du monde réel. En 4, les deux boutons pour choisir le lieu d'exécution du programme (dans le simulateur ou dans l'appartement).

IV – Mise en œuvre de KISS

IV-1 Présentation générale

La figure IV-22 montre la décomposition fonctionnelle de KISS et ses relations de dépendance avec l'infrastructure d'exécution. KISS comprend :

- Une *Grammaire* de description du langage de programmation et les moyens d'éditer cette grammaire pour en ajuster manuellement les unités (noté *Editeur de grammaire* dans le schéma) avec son Interface Homme-Machine, *EdGrIHM*.
- Un éditeur de programmes, noté *EdProg*, et ses deux IHM décrites dans la section III, l'une *IHM.LPN*, l'autre *IHM.Iconic*. Ces IHM font appel à la base de *Programmes existants* pour en proposer la liste à l'utilisateur final qui, dès lors, peut les réutiliser et les éditer (cf. Figure IV-18). À son tour, le noyau fonctionnel d'EdProg comprend un gestionnaire de la *représentation pivot* du langage de programmation (sorte de syntaxe abstraite) et deux *générateurs* qui assurent la traduction entre cette représentation et les deux syntaxes concrètes *LPN* et *Iconic*. Comme dans AutoHan (Blackwell et al. 2002), notre représentation pivot permet d'assurer la propriété multisyntaxe. Notons toutefois que dans la version actuelle de KISS, la traduction entre la représentation pivot et la syntaxe iconique n'est pas bidirectionnelle. En conséquence, l'égale opportunité entre les syntaxes est, pour l'instant, partielle.
- *AnalyseurLx-Sx* assure le rôle usuel d'analyseur lexical et syntaxique. Il utilise évidemment la grammaire, mais de manière plus originale, une base de

connaissances qui permet de répondre à la dynamicité de l'espace intelligent. *AnalyseurLx-Sx* est responsable de la construction de la représentation pivot.

- L'analyseur sémantique *AnalyseurSem* est sollicité par EdProg lorsque l'utilisateur valide la phrase (sur sélection du bouton *Ajouter phrase*). Il effectue une analyse sémantique complémentaire en s'appuyant sur la base de connaissances avant de transmettre le résultat au générateur.
- Le générateur *Générateur*, en s'appuyant sur une grammaire de (des) langage(s) cible(s), assure le rôle usuel de traducteur entre la représentation abstraite d'un programme et une représentation compréhensible par la(les) machines cibles. Dans le cas de CONTINUUM, la machine cible est le Gestionnaire de Contexte. Le résultat de cette traduction est, d'une part, rangé dans la base des programmes existants, et d'autre part transmise à l'infrastructure d'exécution. Nous verrons au chapitre suivant une génération vers un autre langage cible adapté à nos besoins expérimentaux.
- *Simulateur*, qui sert de metteur au point, utilise l'infrastructure cible comme support d'exécution. Il comprend un noyau fonctionnel qui reproduit le fonctionnement du monde réel, et une IHM qui traduit auprès de l'utilisateur final, l'état actuel de ce monde (IHM 2D de la figure IV-19 ou IHM 3D de la figure IV-20).

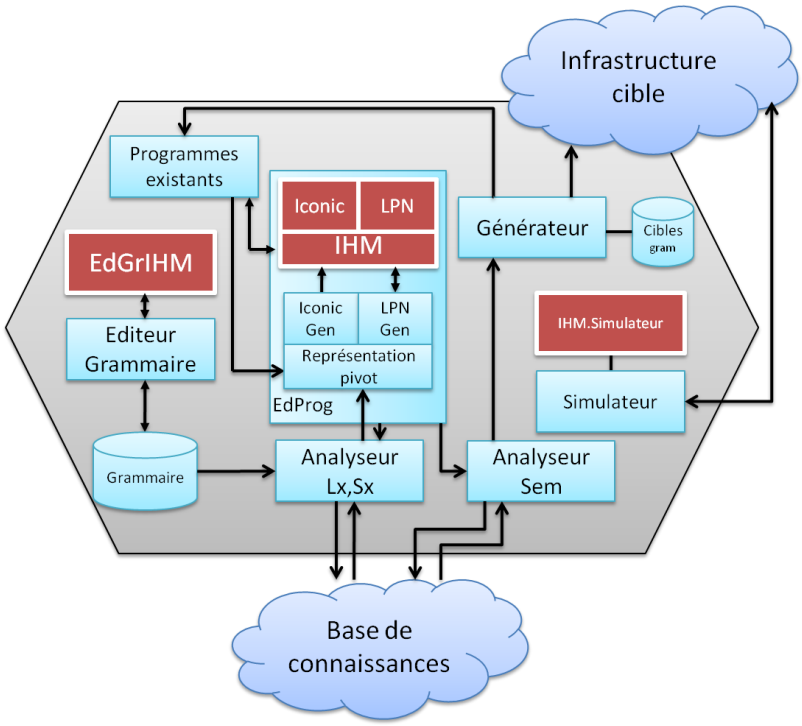


Figure IV-22. Décomposition fonctionnelle de KISS et ses relations de dépendance avec la BdC et l'infrastructure d'exécution. Les boîtes dénotent les composants fonctionnels ; les cylindres, des informations rémanentes ; les flèches, les échanges d'informations et leur direction. La couleur de fond clair indique des noyaux fonctionnels tandis qu'une couleur de fond foncé désigne des IHM.

La figure IV-22 indique les deux points de dépendance de KISS avec son environnement d'exécution : la base de connaissances, dépendante du domaine d'application, et

l'infrastructure qui assure l'adaptation dynamique de l'espace intelligent. La décomposition fonctionnelle de KISS montre comment nous avons minimisé ces dépendances via l'analyseur lexical et syntaxique, le générateur et le simulateur. Dans le cas de l'intégration de KISS dans CONTINUUM (voir figure IV-23) :

- L'analyseur Lx-Sx communique avec la BdC au moyen de requêtes SPARQL.
- Le générateur s'appuie sur la grammaire du langage cible, notée *Ctx-gram*, pour générer les plans d'adaptation sous forme de « situation-AAs ».
- Le simulateur utilise le protocole UPnP de telle sorte qu'un équipement (TV, radio, etc.) virtuel soit un dispositif UPnP. Ainsi, un équipement virtuel est représenté dans l'application par un composant proxy au même titre qu'un équipement réel. Ce faisant, nous obtenons une représentation numérique duale du monde réel.

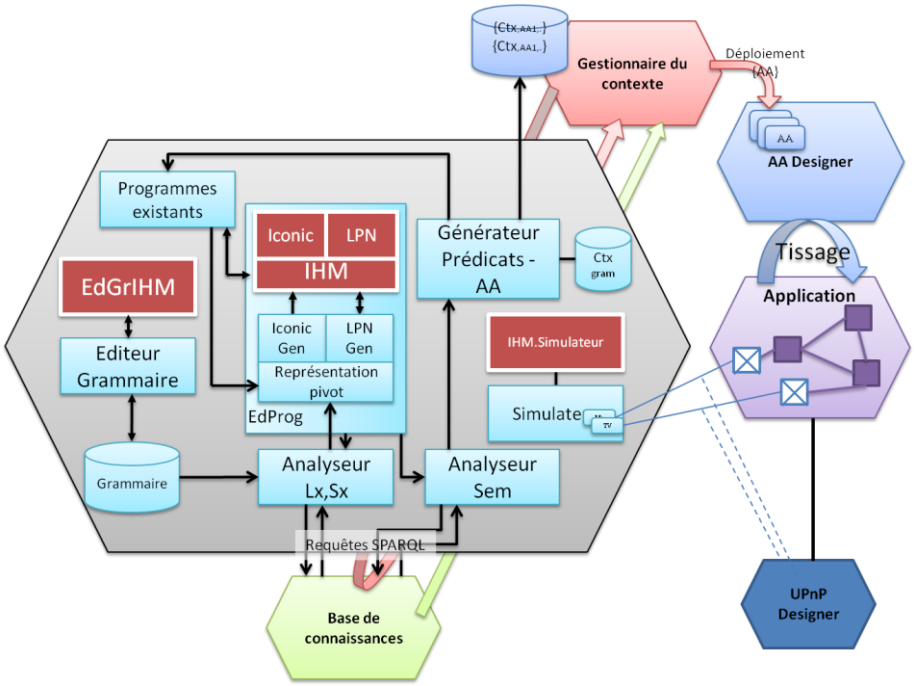
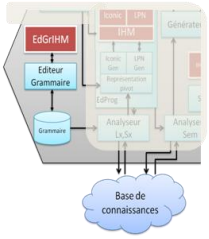


Figure IV-23. KISS installée dans CONTINUUM.

Nous décrivons maintenant en détail le fonctionnement des composants clefs de KISS en suivant les échanges d'information, depuis la grammaire du langage de programmation jusqu'au générateur d'AAs et au metteur point.



IV-2 Grammaire KISS du langage de programmation

La grammaire KISS du langage de programmation est structurée en sous-grammaires de façon à séparer les éléments du langage qui dépendent du domaine de ceux qui n'en dépendent pas. De plus, les éléments dépendants du domaine recouvrent trois aspects qu'il convient de distinguer : les actions possibles dans le domaine (observer, ouvrir, fermer, etc.), les entités du domaine (équipements, personnes, etc.) et les

éléments de contexte (localisation, date, etc.). Nous obtenons ainsi les quatre sous-grammaires suivantes :

- La *grammaire noyau*, indépendante de l'espace intelligent, sert de racine aux trois autres sous-grammaires.
- La *grammaire d'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.
- La *grammaire des classes abstraites* traite de la désignation des entités de l'espace intelligent, ceci indépendamment du contexte.
- La *grammaire du contexte* qui exprime les situations contextuelles.

Les grammaires d'actions, des classes abstraites et du contexte sont enrichies dynamiquement à partir de la BdC. On assure ainsi la souplesse nécessaire à la prise en compte de la dynamicité de l'espace intelligent. En outre, le vocabulaire et la syntaxe sont modifiables au moyen de l'*Editeur de grammaire*. On assure ainsi la personnalisation et l'extensibilité lexicale, syntaxique et sémantique. Toutefois, dans sa version actuelle, EdGrammaire s'adresse à un utilisateur final averti car il nécessite de maîtriser ABNF (Augmented Backus-Naur Form), le métalangage utilisé pour décrire les grammaires.

Nous avons étendu ABNF (Augmented Backus-Naur Form) de deux opérateurs :

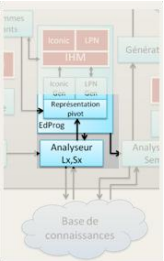
- *Opérateur de redirection* pour désigner une référence à une source extérieure (une autre grammaire ou la BdC).
- *Opérateur sémantique* pour exprimer des traits sémantiques spécifiques au domaine et utilisés par l'analyseur sémantique.

Le tableau IV-1 récapitule l'ensemble des opérateurs de la grammaire KISS. La description complète de la grammaire peut être consultée en annexe IV-1.

Tableau IV-6. Opérateurs utilisés dans la grammaire KISS.

Définition	=	ABNF
Alternative	/	ABNF
Non-terminal	Abz	ABNF
Terminal	" "	ABNF
Groupe	( )	ABNF
Optionnel	[ ]	ABNF
Sémantique	{ }	Extension
Redirection	! !	Extension
Répétition	*	ABNF non utilisé
Intervalle	-	ABNF non utilisé

La section suivante montre comment les éléments de langage de la grammaire KISS sont utilisés par l'analyseur lexical et syntaxique pour construire la représentation pivot et comment cette représentation pivot est utilisée par l'éditeur de programme.



IV-3 Analyseur LX-SX et la représentation pivot

AnalyseurLx-Sx construit la représentation pivot qui modélise, sous forme d'arbre, les phrases permises du langage de programmation. Cette construction se fait en parcourant la grammaire KISS de manière incrémentale (pour suivre à la lettre les choix de l'utilisateur final via l'IHM.LPN et actualiser le feedforward) et fait appel à la BdC pour identifier les entités terminales spécifiques à l'espace intelligent.

Le point d'entrée du parcours de la grammaire est le non-terminal *sentence* de la sous-grammaire noyau. Comme le montre l'extrait du cadre IV-1, *sentence* comprend deux non-terminaux, *condition* et *action*, et porte une décoration sémantique, *{CONDITION:<1> ACTION:<2>}* qui, utilisée par l'analyseur sémantique en vue de la génération, sera détaillée plus loin en IV-5. Nous allons expliquer le fonctionnement de l'analyseur Lx-Sx en détaillant la construction du sous-arbre correspondant à la partie *action* d'une phrase du langage de programmation.

La sous-grammaire noyau stipule qu'une *action* est une suite non vide d'*actionName* reliés par le terminal *et* et qu'un *actionName* est un non-terminal dont la définition se trouve dans la sous-grammaire *actionGram*. À son tour, la sous-grammaire d'actions définit un *actionName* comme une alternative de non-terminaux : *open*, *close*, *watch*, etc. Comme le montre la sous-grammaire d'actions, toutes les définitions de ces non-terminaux commencent par un terminal. L'analyseur Lx-Sx en « sait » suffisamment pour fournir à EdProg la représentation pivot de la figure IV-24 à partir de laquelle EdProg génère le menu déroulant *ouvrir*, *fermer*, *regarder*, etc. de la partie *Je veux*.

Nous pouvons également observer que la grammaire d'actions n'implémente qu'une seule des relations d'Allen, à savoir *start with* (opérateur *et*). Comme l'a montré notre étude de terrain (section III-1-1 du chapitre II), d'autres relations seraient utiles tel que *meet* et *during*. Cependant, ces relations peuvent se révéler difficile à implémenter. Par exemple, *meet* implique que le système soit en mesure de connaitre la fin d'une action, ce qui n'est parfois pas le cas (ex : la café est pret). Ainsi avons nous fait le choix purement technique de nous limiter à la seule relation *start with* dans ce travail de thèse.

Rappelons que la construction de la représentation pivot est incrémentale en réaction immédiate aux choix de l'utilisateur final. Supposons que l'utilisateur choisisse l'élément *regarder*. EdProg colorie, dans la représentation pivot, le chemin correspondant à ce choix, en avertit l'Analyseur Lx-Sx qui poursuit la construction du sous-arbre de *watch*.

Cadre IV-1. Extrait des sous-grammaires noyau et d'actions.

Grammaire noyau	*sentence=condition action {CONDITION:<1> ACTION:<2>}* *action= actionName [operator action]* *actionName=!actionGram!* *operator="et "*
Grammaire d'actions	*actionName= open / close / watch / listen / switchOn / switchOff* *open="ouvrir " opennable {LINK<1>}* *close="fermer " closable {LINK<1>}* *watch="regarder" observable "sur " viewer {<1>LINK<2>}* *listen="ecouter " listenable {LINK<1>}* *switchOn="allumer " switchOnAble {LINK<1>}* *switchOff="eteindre " switchOffAble {LINK<1>}* *opennable=!abstractThingsGram!* *closable=!abstractThingsGram!* *observable=!abstractThingsGram!* *viewer=!abstractThingsGramcom!* *listenable=!abstractThingsGram!* *switchOnAble=!abstractThingsGram!* *switchOffAble=!abstractThingsGram!*

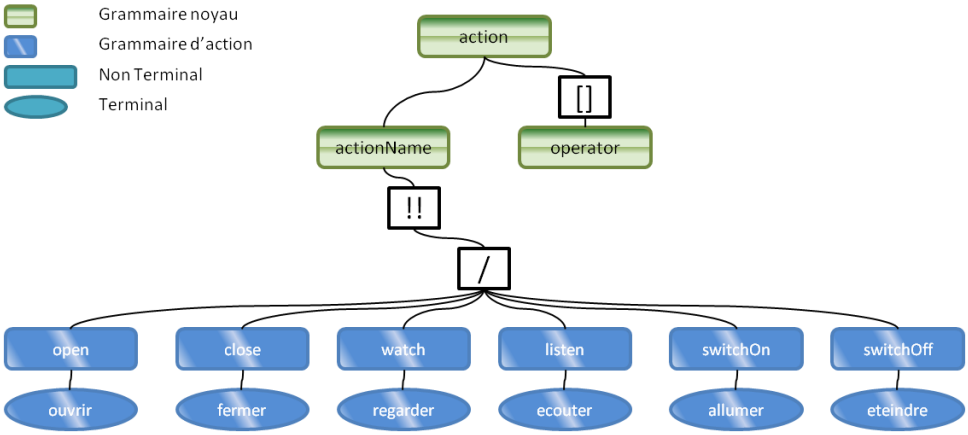


Figure IV-24. Sous-arbre des choix possibles de verbe d'action.

watch est une concaténation du terminal *regarder*, du non-terminal *observable*, du terminal *sur* et du non-terminal *viewer* (ignorons les décorations sémantiques pour l'instant). Il s'agit en effet d'une action qui consiste à regarder « quelque chose, qui est donc observable » et pour regarder cette chose, il faut un viewer, c'est-à-dire un dispositif qui permet de regarder. L'analyseur se doit d'identifier tous les observables possibles de l'espace intelligent, puis, dès que l'utilisateur a fait son choix d'observable, proposer tous les viewers capables de montrer l'observable d'intérêt. C'est là qu'intervient l'ontologie et la BdC. Le non-terminal *observable* fait référence à la sous-grammaire des classes abstraites (Cadre IV-2).

Cadre IV-2. Extrait de la sous-grammaire des classes abstraites.

Grammaire des classes abstraites	observable=!abstractThingsBDC!

Dans la sous-grammaire des classes abstraites, la définition d'*observable*, qui redirige sur la BdC, se traduit par une requête SPARQL. Cette requête demande toutes les instances d'observables présentes dans l'espace intelligent ainsi que les noms de ces instances sous lesquels l'utilisateur final les désigne. Le cadre IV-3 montre, exprimée dans le formalisme ABNF, la réponse de la BdC à cette requête. L'analyseur Lx-Sx complète le sous-arbre selon le schéma de la figure IV-25, en avertit EdProg qui est alors en mesure de proposer un menu à deux éléments : *la température de l'eau*, *l'état de la machine à laver*.

Cadre IV-3. Sous-classes d'observable dans la BdC exprimés en ABNF.

BDC	observable=temperatureObservable / washingMachineStateObservable temperatureObservable="la temperature de l'eau" washingMachineStateObservable="l'état de la machine à laver"

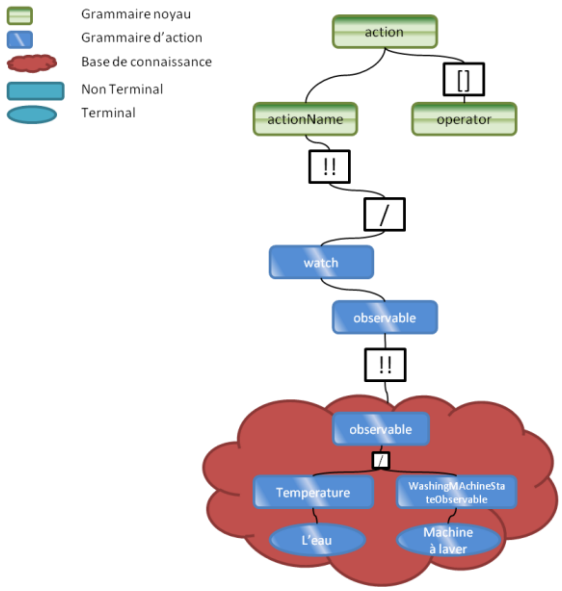


Figure IV-25. Sous-arbre *watch* résultant du choix du terminal *regarder* par l'utilisateur final.

En synthèse, le diagramme de séquence de la figure IV-26 explicite les interactions entre le gestionnaire de la représentation pivot de EdProg, l'analyseur Lx-Sx et la BdC.

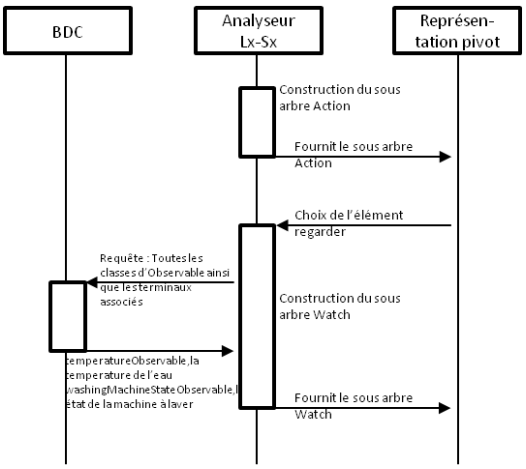
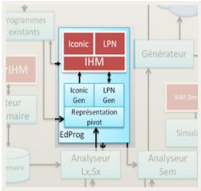


Figure IV-26. Diagramme de séquence UML des interactions entre le gestionnaire de la représentation pivot de EdProg, Analyseur Lx-Sx et la BdC.



IV-4 L'éditeur de programme EdProg

L'éditeur de programme EdProg, rappelons-le, autorise l'utilisation de deux syntaxes concrètes : LPN et visuelle iconique. Comme le montre l'image écran de la figure IV-18, il comprend donc deux IHM (*IHM.LPN* et *IHM.Iconic*) et un noyau fonctionnel constitué de trois composants : le gestionnaire de la représentation pivot et un générateur pour chacune des deux syntaxes concrètes : *LPNGen* et *IconicGen*. Nous avons détaillé dans la section précédente les interactions du gestionnaire de la représentation pivot avec l'analyseur Lx-Sx. Nous complétons ici la description du fonctionnement d'EdProg. Le diagramme de séquence de la figure IV-27 montre les interactions entre les composants du noyau fonctionnel de EdProg, ses deux IHM et l'analyseur sémantique.

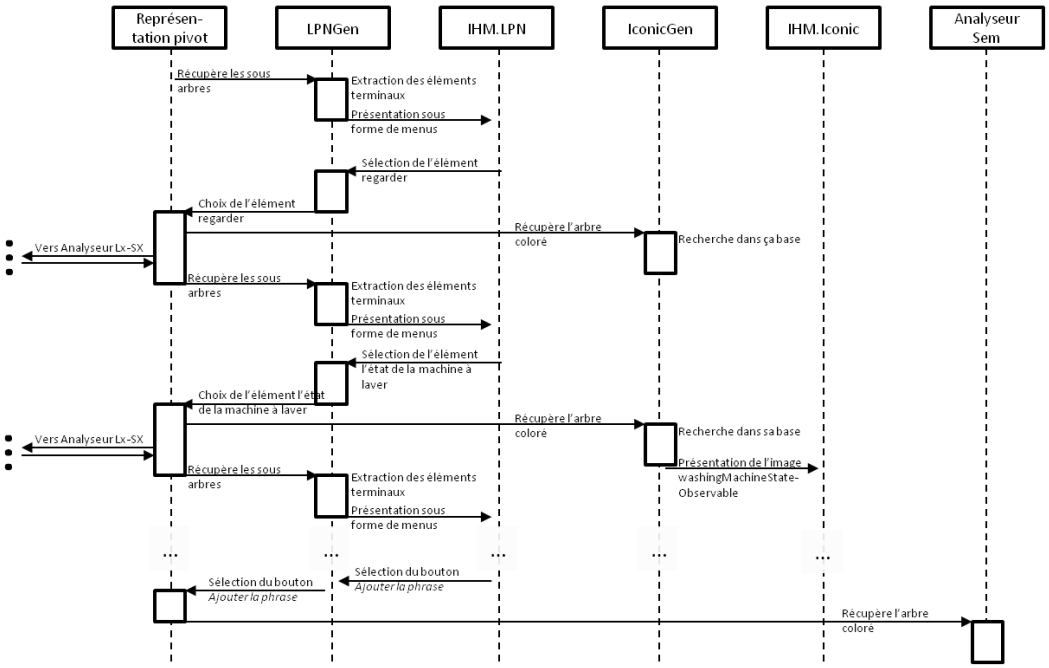


Figure IV-27. Diagramme de séquence UML des interactions entre le gestionnaire de la représentation pivot de EdProg, les générateurs LPNGen et IconicGen, IHM.LPN et IHM.Iconic et AnalyseurSem.

IV-4-1 Générateur LPN et IHM.LPN

Le générateur LPN, *LPNGen*, sert d'intermédiaire entre *IHM.LPN* et le gestionnaire de la représentation pivot. Il récupère auprès du gestionnaire de la représentation pivot, les sous-arbres nouvellement produits par l'analyseur Sx-Lx, en extrait les éléments terminaux pour présentation sous forme de menus par *IHM.LPN*.

En effet, à chaque menu déroulant géré par *IHM.LPN* correspond, dans la représentation pivot, un sous-arbre des choix potentiels permis. *IHM.LPN* gère l'interaction avec l'utilisateur et, sur sélection d'un élément de menu, en avertit *LPNGen*. Ce dernier, qui maintient la correspondance entre les éléments de menu et les feuilles de la représentation pivot, notifie le gestionnaire de la représentation pivot du choix effectué. Cette notification a trois conséquences : le gestionnaire actualise le coloriage des chemins retenus par l'utilisateur, l'analyseur Lx-Sx peut poursuivre le parcours des sous-grammaires et, au nom de l'égale opportunité, *IconicGen* peut actualiser la présentation gérée par *IHM.Iconic*.

À titre illustratif, supposons que le sous-arbre *action* de la représentation pivot soit dans l'état représenté dans la Figure IV-24. Alors, *IHM.LPN* présente le menu de la figure IV-28 et se met à l'écoute de l'utilisateur.

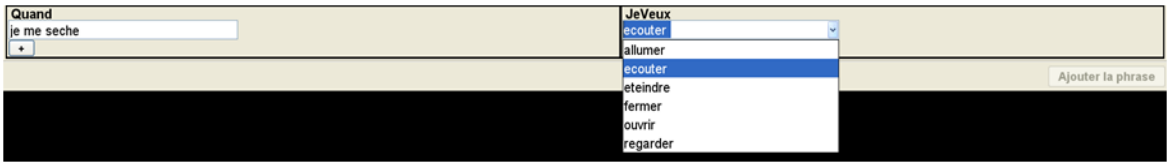


Figure IV-28. À droite, le menu des actions possibles correspondant au sous-arbre de la représentation pivot de la figure IV-24.

Supposons que l'utilisateur sélectionne l'élément *regarder*. L'élaboration de la représentation pivot reprend selon le processus décrit plus haut pour aboutir à l'état de la Figure IV-25. L'IHM en LPN est actualisée dans l'état de la figure IV-29. Notons que le menu des actions reste disponible. Ainsi, l'utilisateur peut revenir sur ses décisions entraînant la réactualisation des coloriages de la représentation pivot et donc celle de l'état de l'IHM de l'éditeur. Notons aussi que la construction d'une phrase en LPN, dirigée par l'analyseur Lx-Sx, est nécessairement syntaxiquement correcte. De plus, l'analyseur faisant appel à la BdC, la phrase est nécessairement sémantiquement correcte.

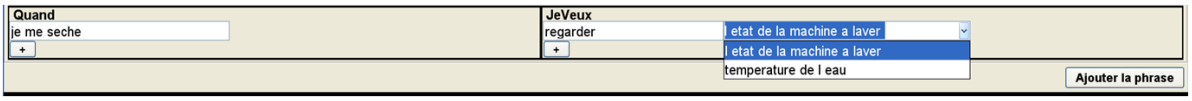


Figure IV-29. L'IHM de l'éditeur en LPN est actualisée avec l'apparition du menu des « observables » présents dans l'espace intelligent suite à la sélection de l'élément « regarder » du menu des actions.

Sur détection de la sélection du bouton *Ajouter la phrase*, *IHM.LPN* avertit, via le *LPNGen*, le gestionnaire de la représentation pivot que la phrase est complète et de là, le gestionnaire fournit l'arbre colorié à *AnalyseurSem*, l'analyseur sémantique. Nous étudions en détail le fonctionnement d'AnalyseurSem dans la section IV-5.

IV-4-2 Générateur iconique et IHM.iconic

Dans leur version actuelle, le générateur iconique *IconicGen* et son IHM, *IHM.Iconic*, sont minimalistes (pour ne pas dire pauvrissimes). Leur présence sert à montrer comment l'architecture d'EdProg assure l'extensibilité de KISS vers du multisyntaxe avec égale opportunité totale.

IconicGen possède une base de couples <image, métadescription> renseignés manuellement au moyen d'un éditeur pour utilisateur averti (cf. figure IV-30). Les métadonnées sont issues de l'ontologie de la BdC. Dans l'exemple de la figure IV-30, l'image du dispositif de nom *Machine à laver* est métadécrite par le nom de la classe de ce dispositif : *washingMachineStateObservable*, sous-classe d'*Observable* dans la BdC.

Illustrons le fonctionnement de *IconicGen* et de son IHM en reprenant l'exemple de la figure IV-29 où l'utilisateur s'apprête à sélectionner *l'état de la machine à laver*. La représentation pivot correspondante est alors celle de la figure IV-25.

Figure IV-30. Ajout dans la base de données de *IconicGen* de la représentation iconique du dispositif *Machine à laver* métadécrit par le nom de classe *washingMachineStateObservable* issu de la BdC.

Sur sélection de l'élément *l'état de la machine à laver*, le processus décrit dans le diagramme de séquence UML de la figure IV-27 est appliqué : le gestionnaire de la représentation pivot, averti de la sélection, diffuse la notification à qui de droit, y compris à *IconicGen*. Ce dernier trouve le non-terminal *washingMachineStateObservable* commun à sa base et au sous-arbre modifié : il fournit à *IHM.Iconic* l'image associée (voir figure IV-31).

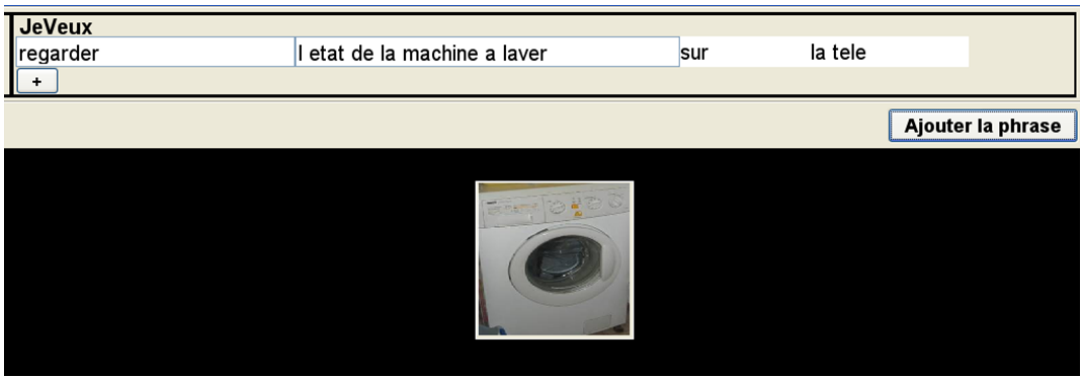
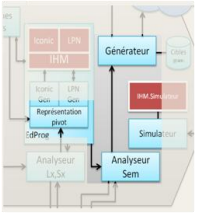


Figure IV-31. Sur sélection de l'élément de menu *l'etat de la machine à laver* dans l'IHM gérée par *IHM.LPN*, la photo du dispositif *Machine à laver* apparaît dans l'IHM de la partie iconique.

Toute phrase validée par l'utilisateur, on le rappelle, a pour effet de solliciter l'analyseur sémantique de KISS.



IV-5 Analyseur sémantique et génération de code

Le rôle central de l'analyseur sémantique, *AnalyseurSem*, est de traiter les décorations sémantiques de l'arbre abstrait correspondant à la phrase que l'utilisateur final vient de valider. Ces traitements doivent faciliter la génération de code par *Générateur* dans le langage cible attendu par le gestionnaire de contexte.

IV-5-1 Principes

AnalyseurSem utilise les opérateurs du cadre IV-4 que nous avons définis pour l'expression des traits sémantiques des sous-grammaires de KISS.

Cadre IV-4. Opérateurs du langage d'expression de la sémantique des non-terminaux des sous-grammaires de KISS.

CONDITION	Dénote la racine du sous-arbre *condition*
ACTION	Dénote la racine du sous-arbre *action*
:	Opérateur d'assignation
<i>	Désignation du $i^{ème}$ non-terminal d'une règle de production
PC	Information complémentaire (par exemple, information de localisation) nécessaire à la génération des futurs Points de Coupe des AAs.
PARAM_STRING, PARAM_INT, PARAM_BOOL	Informations sur les paramètres
EQUAL, SUP, INF	Opérateurs de condition
TOBOOL	Dénote la nécessité de tester l'égalité de ses deux arguments et, si c'est le cas, émettre « vrai »
LASTING	Dénote un aspect duratif (ou progressif)
INST	Dénote un aspect inchoatif (ou instantané)
LINK	Dénote une action de création de liaison

La décoration CONDITION dénote, dans l'arbre abstrait de la phrase à traduire, la racine de l'arbre qui correspond à la description de la situation qui lorsqu'elle se présentera devra provoquer l'exécution d'un plan d'adaptation. Le nœud décoré de ACTION dénote l'arbre correspondant au plan d'adaptation.

<i> permet de désigner des non-terminaux dans les règles de production des sous-grammaires. Par exemple, dans la règle *sentence=condition action {CONDITION :<1> ACTION :<2>}* de la sous-grammaire noyau, <1> désigne le premier non-terminal de la règle, (ici, *condition*) et ce non-terminal correspond à la description d'une situation (opérateur d'assignation « : »).

PC désigne toute information complémentaire nécessaire à la génération de Point de Coupe dans les AAs. Par exemple, dans la règle de production de la grammaire de contexte, *sensorBedroomIn= ''dans la chambre'' {PC location :bedroom},* {PC location:bedroom} précise que les instances de sensorBedroomIn doivent se trouver en un lieu de classe bedroom.

TOBOOL dénote la nécessité de tester, à l'exécution, l'égalité de ses deux arguments et si c'est le cas, émettre un événement « Vrai ». Par exemple, dans la règle de production *presenceDetectorIn= …/…/… {OccupancyStateTOBOOL''Occupied''}, {OccupancyStateTOBOOL''Occupied''}* signifie : si la variable d'état *OccupancyState* des capteurs de présence vaut *Occupied*, alors le booléen Vrai doit être émis.

LINK exprime la création d'une connexion entre ses deux arguments. L'absence du 1er argument indique que l'action est inchoative. Elle est durative s'il y a deux arguments. Nous expliquons ces termes avec les opérateurs LASTING et INST.

Les opérateurs LASTING et INST expriment les aspects *duratif*[40] et *inchoatif* de conditions ou d'actions. Expliquons la distinction par l'exemple, sans entrer dans les subtilités de la linguistique : la condition *quand je rentre dans la chambre* a un caractère instantané. Si une action doit être effectuée en réaction, celle-ci doit débuter dès l'entrée dans la chambre. L'aspect de cette condition est inchoatif. *A contrario*, la condition *quand je suis dans la chambre* a une durée. Si une action doit être effectuée en réaction, celle-ci doit avoir lieu tant que la chambre est occupée. L'aspect de la condition est alors duratif. De même, l'aspect de l'action *allumer la lumière* est inchoactif alors que celui de l'action *regarder la télévision* est duratif (s'il s'agit bien d'être en train de regarder la télévision, non pas juste jeter un regard au poste de télévision auquel cas, l'aspect serait inchoatif).

Il convient donc de considérer quatre combinaisons :

1. Condition et action duratives. Exemple de phrase type : *quand je suis dans la chambre, je veux regarder l'état de la machine à laver sur la télé.*
2. Condition et action inchoatives. Exemple : *quand je rentre dans la chambre, je veux allumer la lumière*.
3. Condition durative, action inchoative. Exemple : *quand je suis dans la chambre, je veux allumer la lumière*.
4. Condition inchoative, action durative. Exemple : *quand je rentre dans la pièce, je veux regarder l'état de la machine à laver sur la télé*.

[40] Les linguistes utilisent aussi le terme progressif.

A partir de ces combinaisons, l'analyseur sémantique doit piloter la génération de prédicats SPARQL et d'AA. Cette génération doit prendre en considération les éléments suivant : (1) une condition inchoative se traduit dans un niveau d'adaptation réflexe, seul garant d'une réactivité conforme à cet aspect ; (2) le greffon d'un AA définit toujours une liaison entre deux composants au minimum.

Par conséquent, la génération répond aux principes suivant (le tableau IV-2 résume ces principes) :

- Une action durative se traduit en AA. En effet, une telle action dispose de deux composants, critère suffisant pour générer un AA.
- Une condition inchoative s'ajoute à la partie gauche de l'AA à générer. Que l'action soit durative ou inchoative, cette condition doit faire partie d'un AA comme source d'événement.
- Une action inchoative met en jeu un seul composant. Il est donc nécessaire de lui associer un second composant pour pouvoir générer un AA. La partie condition, quel que soit son aspect, est alors utilisée comme source d'événement.
- Une condition durative, dans le cas où l'action est durative, se traduit en prédicats SPARQL.
- Dans tous les cas où il manque une condition pour générer un prédicat SPARQL, un prédicat TRUE est généré.

Tableau IV-7. Principes de génération de prédicats SPARQL et d'AAs en fonction de l'aspect de la condition et de l'aspect de l'action.

	Action Durative	Action Inchoative
Condition Durative	AA Durative Prédicats Durative	AA Durative Incho Prédicats TRUE
Condition Inchoative	AA Incho Durative Prédicats TRUE	AA Incho Incho Prédicats TRUE

Nous illustrons la mise en application de ces principes par deux exemples, l'un où condition et action sont inchoatives, l'autre où condition et action sont duratives.

IV-5-2 Exemple 1 : condition et actions inchoatives

L'utilisateur a rédigé la phrase : *Quand je rentre dans la chambre, je veux allumer la lumière de la chambre.* AnalyseurSem produit l'expression sémantique du cadre IV-5. Cette expression sémantique va permettre au générateur de produire l'AA du cadre IV-7 ainsi que les deux prédicats SPARQL du cadre IV-6 qui spécifient la situation dans laquelle cet AA doit être déployé. Le couple <Situation-AA> est ensuite transmis au Gestionnaire de Contexte au format XML.

Cadre IV-5. Expression sémantique produite par AnalyseurSem pour la phrase *Quand je rentre dans la chambre, je veux allumer la lumière de la chambre* : les deux dernières lignes spécifient que les équipements de type *switchOnAbleLight* et *presenceDetectorIn* dont il est question dans la 1ère ligne sont ceux dont la localisation est *bedroom*. La 1ère ligne signifie : produire un événement lorsque l'événement *OccupancyState* de tous les équipements de type *presenceDetectorIn*, vaut *Occupied* et relier cet événement (opérateur *LINK*) à la méthode *SetTarget* (avec le paramètre *True*) de tous les équipements de type *switchOnAbleLight.*

presenceDetectorIn OccupancyStateTOBOOL"Occupied"LINKswitchOnAbleLight "SetTarget" PARAM_BOOL "True"

PC location:bedroom switchOnAbleLight

PC location:bedroom presenceDetectorIn

Cadre IV-6. Ces deux prédicats exprimés en SPARQL spécifient la situation dans laquelle l'AA du cadre IV-7 devra être déployé. À gauche, le lieu d'exécution de la mise en œuvre du AA (monde réel ou monde virtuel du simulateur). À droite, *True* indique que l'AA est toujours valide.

```
ASK ?h WHERE
?x type ModeSelector
?x Mode ?h
FILTRE (?h=real)
```

```
True
```

Cadre IV-7. AA généré correspondant à l'expression sémantique du cadre IV-5. Trouver tous les équipements de type *presenceDetectorIn*, de localisation *bedroom* et présents dans le *monde réel* (*virtual=false*) ; de même, trouver tous les équipements de type *switchOnAbleLight*, de localisation *bedroom*, et présents dans le *monde réel*. Si ces deux ensembles, dénotés respectivement par les variables *presenceDetectorIn* et *swicthOnAbleLight*, ne sont pas vides, alors appliquer l'*advice* de nom *presenceDetectorIn_falseswitchOnAbleLight_11* avec les deux paramètres *presenceDetectorIn* et *switchOnAbleLight*. Cet advice consiste à instancier le composant *bool0*, instance de *WComp.BasicBeans.StringComparator*, un comparateur de chaîne disponible dans WCOMP avec *Occupied* comme valeur de référence. Pour tout élément de l'ensemble *presenceDetectorIn*, (1) créer une connexion entre l'événement *OccupancyState_Evented_NewValue* produit par cet élément et la méthode *Compare* de *bool0*, (2) créer une connexion entre l'événement *MatchedBoolean* produit par *bool0* (si le résultat de la comparaison est vrai), et la méthode *SetTarget* de tous les éléments de *swicthOnAbleLight*. Ce qui a bien pour effet d'allumer la lumière si la chambre est occupée.

```
presenceDetectorIn = *?type=presenceDetectorIn&location=bedroom&virtual=false
switchOnAbleLight  = *?type=switchOnAbleLight&location=bedroom&virtual=false
advice presenceDetectorIn_falseswitchOnAbleLight_11(presenceDetectorIn,
switchOnAbleLight) :

bool0:WComp.BasicBeans.StringComparator(ReferenceValue:="Occupied")

presenceDetectorIn.^OccupancyState_Evented_NewValue -> (bool0.Compare)

bool0.^MatchedBoolean -> (switchOnAbleLight. SetTarget )
```

Nous allons maintenant décrire comment *AnalyseurSem* aboutit à l'expression sémantique du cadre IV-5. Il faut pour cela partir de l'arbre abstrait de la phrase source que EdProg a transmis à AnalyseurSem (voir figure IV-32). Il faut également exploiter les traits sémantiques des non-terminaux de l'arbre au moyen des sous-grammaires de KISS (cadres IV-8, IV-9, IV-10).

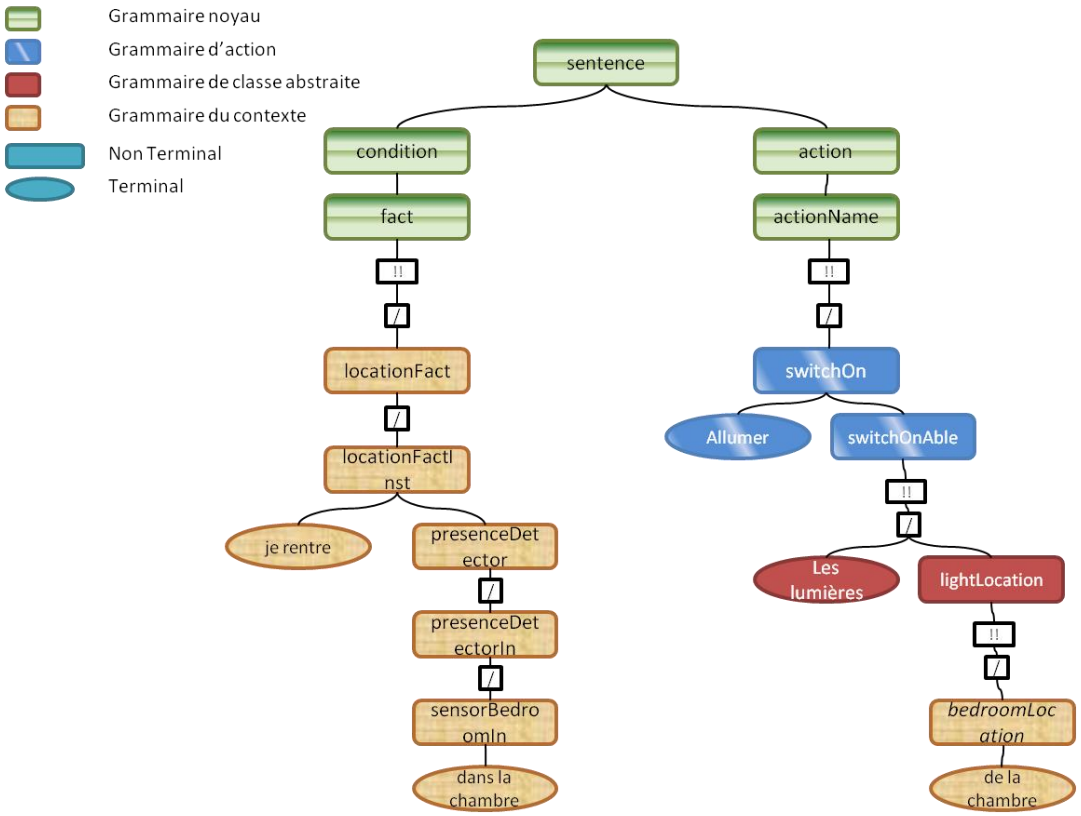


Figure IV-32. Arbre abstrait correspondant à la phrase : *quand je rentre dans la chambre, je veux allumer la lumière de la chambre*.

Cadre IV-8. Extrait de la grammaire du contexte.

Grammaire du contexte	fact=locationFact / timeFact locationFact=locationFactLasting / locationFactInst locationFactInst="je rentre " presenceDetector {INST<1>} presenceDetector=presenceDetectorIn / presenceDetectorOut presenceDetectorIn= sensorBathroomIn / sensorBedroomIn / sensorKitchenIn {OccupancyStateTOBOOL"Occupied"} sensorBedroomIn="dans la chambre" {PC location:bedroom}

Cadre IV-9. Extrait de la grammaire des classes abstraites.

Grammaire de classes abstraites	*switchOnAble=switchOnAbleLight / switchOnAbleCoffee* switchOnAbleLight="les lumieres " [lightLocation] [lightIntensity] {"SetTarget" PARAM_BOOL "True"} lightLocation=!contextualInfoGram!

Cadre IV-10. Extrait de la grammaire du contexte pour le non-terminal lightLocation

Grammaire du contexte	*lightLocation=bathroomLocation / bedroomLocation / kitchenLocation* bathroomLocation="de la salle de bain " {PC location:bathroom} bedroomLocation="de la chambre " {PC location:bedroom} kitchenLocation="de la cuisine " {PC location:kitchen}

AnalyseurSem applique un algorithme à 3 étapes, d'abord pour la partie *condition*, puis pour la partie *action* de l'arbre : (1) recherche de la sémantique générale de la partie en cours d'analyse et de son aspect inchoatif/duratif, (2) recherche de traits

complémentaires (avec les opérateurs PC pour des informations complémentaires sur des objets, PARAM_STRING, PARAM_INT et PARAM_BOOL pour les informations sur les paramètres), (3) recherche des ports pour la création de futures connexions.

Commençons par le sous-arbre *condition* de la figure IV-32, *quand je rentre dans la chambre*.

étape 1. Le non-terminal *locationFactInst* donne la réponse à la première étape : dans la grammaire du cadre IV-8, la règle de production *locationFactInst=''je rentre'' presenceDetector {INST <1>}*, {INST <1>} indique que l'on a affaire à une condition inchoative. Le résultat de cette 1ère étape est :

INSTpresenceDetectorIn

étape 2. AnalyseurSem cherche ensuite à acquérir les informations complémentaires (opérateur PC) pour chaque non-terminal du sous-arbre *condition* et cela en relation avec les sous-grammaires. Seul le non-terminal *sensorBedroom* possède un trait sémantique : *{PC location:bedroom}*. Le résultat de la 2ème étape est maintenant :

INSTpresenceDetectorIn

PC location:bedroom presenceDetectorIn

étape 3. Cette étape consiste à trouver les ports pour établir les futures connexions. AnalyseurSem procède comme précédemment à parcourir les non-terminaux de l'arbre abstrait et les sous-grammaires, mais cette fois en repérant les ports. Seul le non-terminal *presenceDetectorIn* répond à la recherche avec *OccupancyStateTOBOOL"Occupied"*. L'analyse de la partie condition de la phrase est maintenant complète :

INSTpresenceDetectorIn OccupancyStateTOBOOL"Occupied"

PC location:bedroom presenceDetectorIn

Le même algorithme à trois phases (sémantique générale, informations complémentaires, ports) est appliqué à la partie *action* de la figure IV-32, *je veux allumer la lumière de la chambre*.

étape 1. Cette étape s'appuie sur le non-terminal *switchOn* (cf. la sous-grammaire du cadre IV-1). On obtient :

LINKswitchOnAbleLight

étape 2. Après la recherche des informations PC, le résultat de la 2ème étape produit :

LINKswitchOnAbleLight

PC location:bedroom switchOnAbleLight

étape 3. Dans la sémantique déjà extraite, il n'y a qu'un seul non-terminal impliqué, donc un seul port de communication.

LINKswitchOnAbleLight "SetTarget" PARAM_BOOL "True"

PC location:bedroom switchOnAbleLight

L'opérateur *PARAM_BOOL* a deux arguments : l'argument de gauche désigne un nom de méthode à un paramètre booléen ; l'argument de droite est la valeur du paramètre d'appel de cette méthode. Ici la méthode est *SetTarget,* elle prendra comme paramètre *True*.

Nous constatons que l'opérateur LINK n'a pas de source, ce qui est contraire à la sémantique d'une liaison. Dans ce cas, l'analyseur sémantique utilise l'expression sémantique de la partie condition comme source de l'opérateur LINK alors que cette partie condition devrait servir à la description de la situation. La situation devient true (partie droite du cadre IV-6), sa description devenant une source événementielle d'AA. On obtient au final :

presenceDetectorIn OccupancyStateToBOOL"Occupied"LINKswitchOnAbleLight "SetTarget" PARAM_BOOL "True"

PC location:bedroom switchOnAbleLight

PC location:bedroom presenceDetectorIn

IV-5-3 Exemple 2 : condition et action duratives

L'utilisateur a rédigé la phrase : *Quand je suis dans la chambre, je veux regarder l'état de la machine à laver sur la télé.* AnalyseurSem produit l'expression sémantique du cadre IV-11 qui aide le générateur à produire l'AA du cadre IV-13 ainsi que les deux prédicats SPARQL du cadre IV-12 qui spécifient la situation dans laquelle cet AA doit être déployé.

Cadre IV-11. Expression sémantique produite par AnalyseurSem pour la phrase *Quand je suis dans la chambre, je veux regarder l'état de la machine à laver sur la télé* : La 1ère ligne décrit la situation qui nécessitera une adaptation. La 2ème ligne spécifie que les *presenceDetectorIn* dont il est question doivent se trouver dans une chambre. Ces deux lignes conduisent à la génération du prédicat de droite du cadre IV-12. La dernière ligne de l'expression sémantique se lit comme suit : pour tout équipement qui déclare dans ses métadonnées être un *washingMachineObservable*, écouter sa variable événementiel *RemainingTime.* Quand cette variable change, la transférer aux équipements qui déclarent dans leurs métadonnées être des *display* via leur méthode *setValue.* Cette ligne permettra de générer l'advice de l'AA du cadre IV-12.

LASTINGpresenceDetectorIn OccupancyStateTOBOOL"Occupied"

PC location:bedroom presenceDetectorIn

washingMachineObservable RemainingTime LINKdisplay SetValue

Cadre IV-12. Les deux prédicats exprimés en SPARQL spécifient la situation dans laquelle l'AA du cadre IV-13 devra être déployé. À gauche, le lieu d'exécution de la mise en œuvre du AA (ici, monde réel). À droite, la chambre doit être occupée.

```
ASK ?h WHERE
?x type ModeSelector
?x Mode ?h
FILTRE (?h=real)
```

```
ASK ?h WHERE
?x type presenceDetector
?x location bedroom
?x OccupancyState ?h
FILTRE (?h=Occupied)
```

Cadre IV-13. AA généré correspondant à l'expression sémantique du cadre IV-11. Trouver tous les équipements de type *washingMachineObservable* et présents dans le *monde réel* (*virtual=false*) ; de même, trouver tous les équipements de type *display* et présents dans le *monde réel*. Si ces deux ensembles, dénotés respectivement par les variables *washingMachineObservable* et *display*, ne sont pas vides, alors appliquer l'*advice* de nom *washingMachineObservable_falsedisplay_11* avec les deux paramètres *washingMachineObservable* et *display*. Cet advice consiste à créer pour tout élément de *washingMachineObservable* une connexion entre l'événement *RemainingTime_Evented_NewValue* produit par cet élément et la méthode *SetValue* de tous les éléments de *display*.

```
washingMachineObservable = *?type=washingMachineObservable&virtual=false
display = *?type=display&virtual=false
advice washingMachineObservable_falsedisplay_11(washingMachineObservable, display) :
washingMachineObservable.^RemainingTime_Evented_NewValue -> (display.SetValue)
```

Comme précédemment, *AnalyseurSem* aboutit à l'expression sémantique du cadre IV-11 à partir de l'arbre abstrait de la phrase source (figure IV-33) et en s'appuyant sur les traits sémantiques des non-terminaux au moyen des sous-grammaires de KISS (cadres IV-14, IV-15, IV-16.

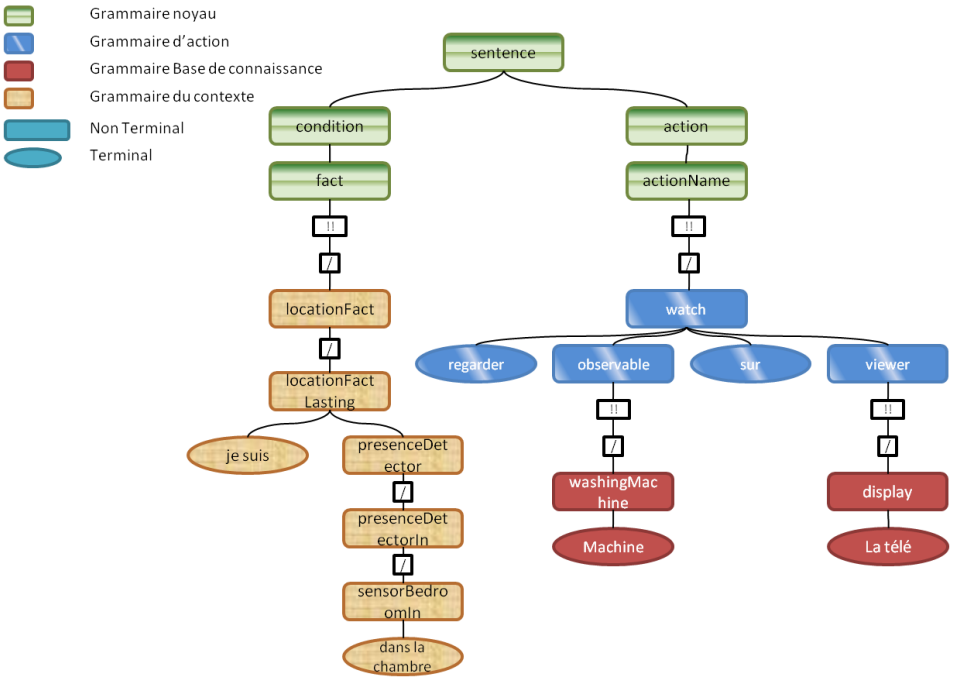


Figure IV-33. Arbre abstrait correspondant à la phrase : « quand je suis dans la chambre, je veux regarder l'état de la machine à laver sur la télé ».

Analyse sémantique de la partie *condition* de l'arbre abstrait de la figure IV-33, *quand je suis dans la chambre.*

étape 1. Le non-terminal concerné est *locationFactLasting*, dont l'aspect, d'après la règle de production de la grammaire de contexte (cadre IV-14) est duratif (*LocationFactLasting="je suis" presenceDetector {LASTING<1>}*. Le résultat est donc :

LASTINGpresenceDetectorIn

étape 2. Recherche des informations complémentaires. La règle de production *sensorBedroomIn="dans la chambre" {PC location:bedroom}* (cadre IV-14) conduit au résultat suivant :

LASTINGpresenceDetectorIn

PC location:bedroom presenceDetectorIn

étape 3. Recherche des ports. La grammaire du cadre IV-14 nous offre une seule proposition avec la règle de production *presenceDetectorIn= sensorBathroomIn / sensorBedroomIn / sensorKitchenIn {OccupancyStateTOBOOL"Occupied"}* ce qui donne au final pour la partie condition :

LASTINGpresenceDetectorIn OccupancyStateTOBOOL"Occupied"

PC location:bedroom presenceDetectorIn

En français, cette expression s'écrit : pour tous les équipements qui déclarent dans leur métadonnées être des *presenceDetectorIn* et se trouver dans *bedroom*, écouter leur variable événementiel *OccupancyState.* Quand elle vaut *Occupied*, émettre *vrai*.

Cadre IV-14. Extrait de la grammaire du contexte.

Grammaire du contexte	fact=locationFact / timeFact locationFact=locationFactLasting / locationFactInst locationFactLasting="je suis " presenceDetector {LASTING<1>} presenceDetector=presenceDetectorIn / presenceDetectorOut presenceDetectorIn= sensorBathroomIn / sensorBedroomIn / sensorKitchenIn {OccupancyStateTOBOOL"Occupied"} sensorBedroomIn="dans la chambre" {PC location:bedroom}

Analyse sémantique de la partie *action* de l'arbre abstrait de la Figure IV-33, *je veux regarder l'état de la machine à laver sur la télé*. Nous allons nous servir des extraits des grammaires des cadres IV-15 et IV-16.

Cadre IV-15. Extrait de la grammaire BdC.

Grammaire BdC	observable=temperatureObservable / washingMachineStateObservable temperatureObservable="la temperature de l'eau" {CurTemp} washingMachineStateObservable="l'état de la machine à laver" {RemainingTime}

Cadre IV-16. Extrait de la grammaire des sous-classes abstraites.

Grammaire des classes abstraites	viewer=display display="la tele " {SetValue}

étape 1. Le non-terminal concerné est *watch* défini par la règle de production du cadre IV-1 : *watch="regarder" observable "sur " viewer {<1>LINK<2>})*. Le résultat est donc :

washingMachineObservableLINKdisplay

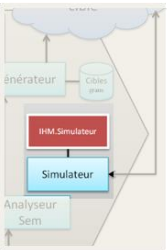
étape 2. Recherche des informations complémentaires (opérateurs PC, PARAM_INT, PARAM_STRING, PARAM_BOOL). Les sous-grammaires n'indiquent aucune information complémentaire. Le résultat reste :

washingMachineObservableLINKdisplay

étape 3. Recherche des ports des entités washingMachineObservable et display entre lesquels établir la liaison. Pour *washingMachineObservable*, il faut demander à la BdC (cf. cadre IV-15), pour *display*, l'information se trouve spécifiée dans la grammaire des sous-classes abstraites (cadre IV-16). Ce qui donne au final pour la partie action :

washingMachineObservable RemainingTime LINKdisplay SetValue

Ou, en français : Pour tous les équipements qui déclarent dans leur métadonnées être des *washingMachineObservable*, écouter leur variable événementiel *RemainingTime.* Quand cette variable change, la transférer aux équipements qui déclarent dans leurs métadonnées être des *display* via leur méthode *setValue*.



IV-6 Simulateur, metteur au point

Le monde simulé permet à l'utilisateur de vérifier que le programme qu'il vient de concevoir est conforme à ses attentes, avancer ou reculer dans le temps à la vitesse désirée comme indiqué dans la section III. Le moteur du monde simulé doit :

- Fournir une représentation virtuelle de l'environnement réel dans lequel évolue l'utilisateur.
- Permettre à l'utilisateur de déplacer un avatar qui le représente virtuellement.
- Permettre l'ajout dynamique d'équipements virtuels, existence duale des équipements physiques du monde réel.
- Permettre à l'utilisateur d'interagir avec les équipements du monde simulé.

Le tableau IV-3 reprend les différents critères de choix. Pour la représentation virtuelle 3D, notre choix s'est porté sur le moteur de jeu Blender. Succinctement, Blender permet d'éditer directement le monde simulé dans son moteur de rendu. La communauté utilisateurs est très active, ce qui facilite l'effort de développement, et surtout, Blender utilise des scriptes Python qui permettent notamment d'ajouter facilement un mécanisme de communication.

Sur le plan mise en œuvre, un équipement réel ou simulé est un service UPnP. La seule distinction parmi ces dispositifs réside dans leurs métadonnées. Un dispositif virtuel aura dans ses métadonnées : *virtual=true* alors que pour un dispositif du monde réel *virtual=false.*

Tableau IV-3. Comparaison de différents moteurs de monde virtuels.

	DiaSuite	OpenSim	Blender	jMonkeyEngine
Dimension maximale supportée	2D	3D	**3D**	3D
Qualité de la documentation du code source	Code source non fourni	Insuffisante	Faible	Médiocre
Codé en	Java	C Sharp	Python	Java
Langage de scripting	Java	LSL	**Python**	Java
Doc. pour l'utilisation du logiciel/API	Bonne	Médiocre	**Bonne**	Bonne
Editeur de modèles 3D	Non	Non	**Oui**	Non
Moteur physique	Non	Oui	Oui	Oui
Pilotage de l'extérieur	Facile	Requiert la modification du code source	Possible	Facile
Séparation des entités	Oui	Oui	Oui	Oui
Relation entre les entités	Oui	Oui	Oui	Oui
Importation de modèles 3D et des animations	Non	Limité	**Oui**	Oui
Communauté	Inexistante	Faible	Active	Active
Définition et animation des gestes	Non	Non	**Oui**	Non
Extensible	Non	Non	**GUI et moteur de jeu extensible**	Probablement oui

Le monde simulé peut également être utilisé comme moyen de piloter le monde réel. Dans ce cas, les équipements présents dans le monde simulé sont directement connectés avec leurs homologues réels. Ce mode de connexion se passe de l'infrastructure cible, et ne rentre pas dans le cadre de la programmation d'habitat intelligent par l'utilisateur.

V – Conclusion

Nous venons de décrire KISS et son infrastructure d'exécution, CONTINUUM. À cette occasion, nous avons introduit le concept de point de contrôle qui, au sein d'un espace intelligent, permet d'ouvrir la boucle d'adaptation à l'utilisateur. Nous avons identifié trois points de contrôle pour chacun des niveaux d'adaptation : réflexe, tactique et stratégique avec, en regard les méta-IHM idoines.

Nous avons décrit un exemple de méta-IHM réflexe qui permet à l'utilisateur de reconfigurer par le geste la distribution de l'interface homme-machine d'un visualisateur de photos. Nous avons montré comment cette application patrimoniale a pu être enrichie sur le plan interactionnel sans modifier le code original.

Dans notre terminologie, KISS est un DUF qui joue le rôle de méta-IHM tactique. Les points originaux clefs de KISS incluent : une approche multisyntaxe (avec égale opportunité partielle dans sa version actuelle), un langage LPN avec menus contextuels en complète conformité avec l'état de l'espace intelligent, une génération de code, non pas de code classique, mais de plans d'adaptation compréhensibles par une infrastructure à composants orientés service et de là, une fusion des phases de conception et d'exécution, exécution qui peut avoir lieu dans un monde dual numérique simulé comme dans le monde réel.

Ces choix ont été faits en réponse à certaines lacunes de l'état de l'art en matière de PUF-DUF-GLUF. En particulier, nous sommes les seuls avec TeC à pouvoir fusionner les phases de conception et d'exécution. Comme pour A Cappella, iCap et Pantagruel, nous proposons d'outils de mises au point dans les mondes physique et numérique. Comme pour Pantagruel et Tec, nous proposons de programmer par composition mais permettons aussi, comme le font iCap, CAMP et AutoHan, de construire des programmes ex-nihilo. Il convient maintenant de valider nos choix par l'expérimentation avec l'utilisateur final. C'est l'objet du chapitre suivant.

Chapitre V : L'acceptabilité de KISS

Table des matières

Avant-propos 134

I – Préparatifs expérimentaux 135
I-1 Influence du scénario 136
I-1-1 Adaptation de l'extrait de scénario 136
I-1-2 Ajout d'équipements dans DOMUS 137
I-1-3 Utilisation de la technique du Magicien d'Oz 137
I-2 Influence de l'infrastructure d'accueil 139
I-2-1 Extension logicielle des équipements UPnP 139
I-2-2 Remplacement de WCOMP par un intergiciel dédié 139
I-2-3 Simulation de la Base de Connaissances (BdC) 141
I-2-4 L'habitat virtuel et le metteur au point 141

II – Protocole expérimental 142
II-1 Déroulement de l'expérimentation 143
II-2 Recrutement des participants 145

III - Résultats du test d'acceptabilité 145
III-1 Résultats relatifs à notre approche 146
III-1-1 Aspects positifs 146
III-1-2 Difficultés rencontrées 148
III-1-3 Pistes d'améliorations 151
III-2 Résultats relatifs à KISS 152
III-2-1 Aspects positifs 152
III-2-2 Difficultés rencontrées 153
III-2-3 Pistes d'améliorations 153

IV – Conclusion 155

Résumé

Ce chapitre décrit l'expérimentation que nous avons réalisée pour valider notre approche et évaluer l'acceptabilité de KISS. Afin de concilier pragmatisme et réalisme expérimental, nous avons proposé à treize participants représentatifs de programmer dans DOMUS, un habitat intelligent, une séquence d'activités routinières : le lever matinal, identifié lors de notre étude de terrain amont comme l'un des moments clefs de la vie quotidienne. L'analyse des résultats montre que les utilisateurs adhèrent aux principes de KISS, apprécient l'utilisation du langage pseudonaturel, de même

l'utilisation de la réplique virtuelle de l'habitat. Les difficultés rencontrées portent pour l'essentiel sur un vocabulaire jugé trop limité. Un mécanisme de co-construction système-utilisateur est clairement souhaitable, de même le multisyntaxe accompagné de programmation *sur* et *avec* exemples.

Avant-propos

Nous venons de décrire KISS selon les deux perspectives complémentaires, utilisateur final et implémentation. En synthèse, KISS permet à l'utilisateur de rédiger des programmes en langue française pseudonaturelle et d'en tester l'exécution dans un habitat intelligent réel ou dans son dual virtuel. Sur le plan technique, KISS est l'un des services de l'infrastructure d'exécution CONTINUUM qui, à son tour, s'appuie sur l'intergiciel WCOMP pour la composition et la reconfiguration dynamiques. Composition et reconfiguration sont exprimées en Aspect d'Assemblage (AA). KISS, qui joue le rôle de méta-IHM tactique dans l'infrastructure CONTINUUM, traduit les programmes de l'utilisateur en plans d'adaptation sous la forme de couples « situation-AA ». Rappelons que les équipements gérés par WCOMP doivent être « compatibles UPnP ».

Ce chapitre porte sur la dernière étape de notre recherche doctorale : la validation de KISS par l'évaluation de son acceptabilité en environnement réaliste (voir figure V-1). Les deux choix suivants visent à concilier pragmatisme et réalisme expérimental :

- Retenir un extrait du scénario établi et validé par notre étude de terrain en tout début de thèse : en l'occurrence, la séquence du réveil matinal (cf. chapitre II).
- Installer KISS dans un appartement intelligent aux équipements « compatibles UPnP » : en l'occurrence, le plateau d'expérimentation DOMUS du LIG, équipé de capteurs et de dispositifs communicants contrôlables, mais aussi doublé d'une régie (Gallissot 2012).

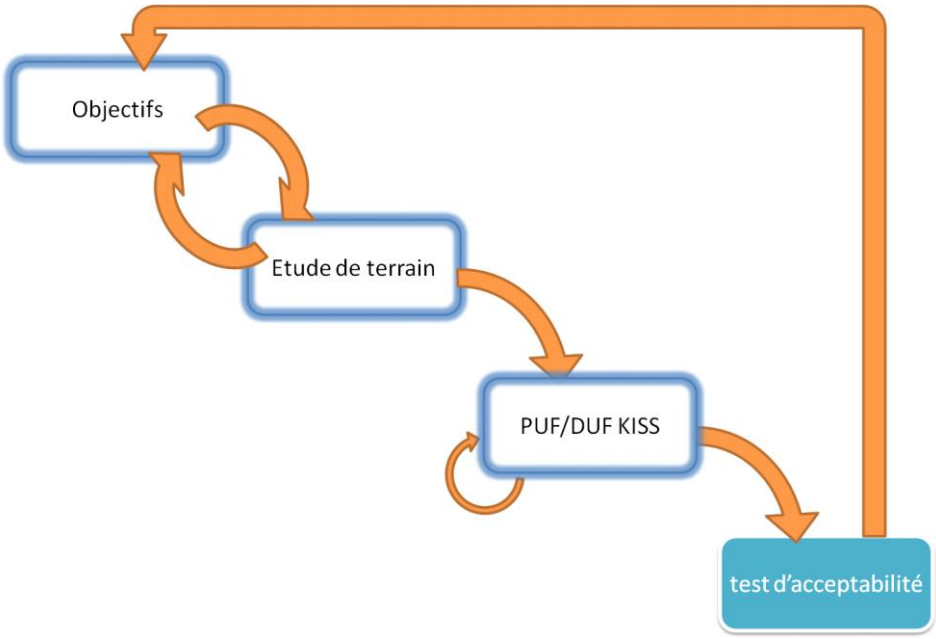


Figure V-1. Test d'acceptabilité pour évaluer la solution KISS et ouvrir de nouvelles perspectives.

Ce chapitre comprend trois parties : la première précise les adaptations nécessaires à la réalisation des tests et les compromis réalisés entre la faisabilité technique dans le temps imparti et le respect du requis de qualité « écologique ». Puis, nous présentons le protocole expérimental (section II) suivi, en section III, de l'analyse des résultats.

I – Préparatifs expérimentaux

Comme le montre le plan de la figure V-2, DOMUS est un appartement entièrement meublé. Il est équipé de capteurs et de dispositifs communicants contrôlables. Certains de ces équipements sont natifs UPnP. C'est le cas par exemple des équipements audio. D'autres, comme les volets, les lumières, ont nécessité des ajustements pour les rendre compatibles UPnP[41].

Bien qu'il ait été conçu pour mener des expérimentations sur l'habitat intelligent, DOMUS n'est toutefois pas directement « Plug and Play ». Nous avons donc décliné les préparatifs expérimentaux en deux volets : (1) Vérification de la faisabilité de l'extrait de scénario dans DOMUS. (2) Vérification de l'intégration technique de KISS et de CONTINUUM/WCOMP dans DOMUS.

Selon toute attente, il a fallu faire des adaptations à la fois du scénario et de la plate-forme d'accueil, motivées par un bon compromis entre pragmatisme et réalisme.

[41] Ces ajustements ont été réalisés par Mathieu Galissot (Gallissot 2012).

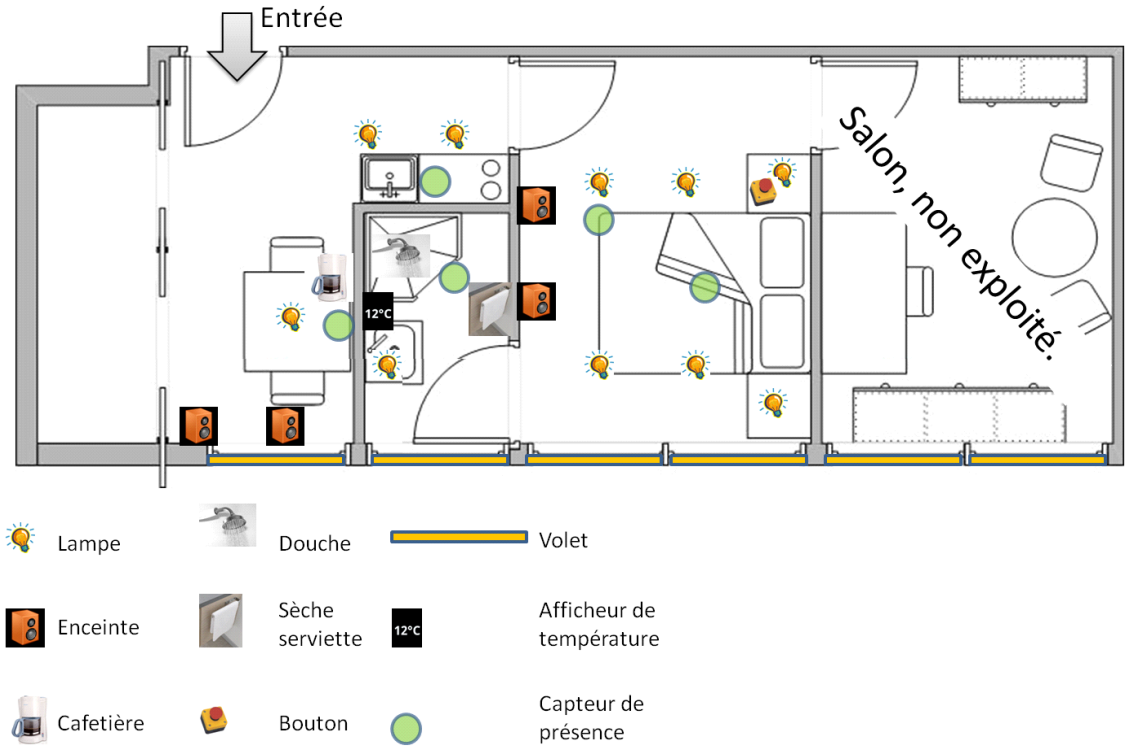


Figure V-2. Plan de DOMUS avec implantation des différents équipements référencés dans notre test. Le salon n'est pas référencé dans notre protocole expérimental.

I-1 Influence du scénario

Le scénario original peut être consulté dans sa totalité dans l'Annexe II-3. De façon à limiter la durée des passations tout en respectant le requis de réalisme, nous en avons extrait une séquence pertinente : le lever matinal qui relève d'activités routinières, activités auxquelles l'intelligence ambiante est susceptible d'apporter des améliorations (Coutaz et al. 2010), (Davidoff et al. 2012). Afin de nous assurer de la faisabilité de cette séquence dans DOMUS, nous avons vérifié le fonctionnement de chaque service et équipement référencés dans l'extrait de scénario. Ces vérifications systématiques ont appelé trois types d'actions correctives : l'adaptation de l'extrait de scénario, l'ajout d'équipements dans DOMUS et l'introduction d'un Magicien d'Oz.

I-1-1 Adaptation de l'extrait de scénario

Nous avons adapté le scénario du lever matinal en y apportant deux suppressions, une modification et un ajout.

1) L'extrait original fait référence aux toilettes : *il prend le chemin des toilettes*. DOMUS ne disposant pas de toilettes, nous avons dû supprimer cette référence.

2) L'extrait original fait référence à la télévision : *Allumer le téléviseur*. DOMUS possède un téléviseur dont le pilotage via UPnP était alors en cours d'implémentation. Nous avons supprimé la référence à cet équipement au regard du requis de robustesse.

3) L'extrait original fait référence à la radio de la salle de bain alors qu'il n'y en a pas dans celle de DOMUS. Or DOMUS est un petit appartement de 40m² environ aux cloisons fines. Il s'ensuit que le son joué dans la chambre s'entend tout aussi bien depuis la salle de bain. Nous avons modifié l'extrait du scénario en conséquence.

4) Comme DOMUS dispose de volets UPnP et que la séquence se déroule le matin au réveil, nous avons jugé pertinent d'ajouter l'ouverture des volets au lever du lit.

Enfin, l'extrait de scénario a été simplifié et reformulé de façon à ce que les utilisateurs participant au test d'acceptabilité puissent se l'approprier facilement. Le texte final de cet extrait est présenté dans la figure V-8.

I-1-2 Ajout d'équipements dans DOMUS

L'extrait de scénario fait référence à un indicateur lumineux de la température de l'eau : *une lumière projetée sur le plafond au-dessus de la douche lui indique la température de l'eau*. Cette phrase nécessite la présence d'un capteur de température et d'un afficheur que nous avons intégrés dans DOMUS (Figure V-3). L'intérêt de cet ajout est de rendre possible l'expression de règles comprenant des verbes d'aspect duratif. Dans *je veux regarder la température de l'eau sur la douche*, le verbe d'action "regarder" est d'aspect duratif alors que les verbes utilisés dans l'extrait de scénario sont d'aspect inchoatif comme dans *allumer la lumière*, *ouvrir les volets*.

Figure V-3. L'afficheur de température réalisé avec un SmartPhone.

I-1-3 Utilisation de la technique du Magicien d'Oz

La technique du Magicien d'Oz consiste à faire simuler par un compère humain les fonctions manquantes d'un système de telle sorte que l'utilisateur croie que ces fonctions sont véritablement assurées par le système (Dahlback et al. 1993). L'existence du compère, un expérimentateur averti, est inconnue de l'utilisateur avant et pendant les passations.

Dans notre extrait de scénario, certains comportements de l'habitat sont conditionnés par l'existence de capteurs dont DOMUS ne dispose pas, ou bien ses capteurs sont de fiabilité insuffisante. Par exemple, la lumière doit s'allumer quand on rentre dans une pièce et s'éteindre quand il n'y a personne. Il s'avère que les capteurs de DOMUS détectent le mouvement, non pas la présence. En conséquence, la présence d'une personne en mouvement est détectée et signalée immédiatement. En revanche,

l'absence de cette personne, qui devient vraie lorsqu'elle sort de la pièce, est signalée par DOMUS au bout d'une trentaine de secondes. Ce délai est pénalisant. De même, le porte-serviettes, le lit et le bouton starter sont, soit équipés de capteurs peu fiables, soit ne sont pas instrumentés. Profitant de l'existence d'une régie, nous avons implémenté un service de Magicien d'Oz[42] (MOz) pour assurer correctement et de manière fiable les fonctions manquantes de DOMUS.

La régie du plateau d'expérimentation, qui inclut des caméras et des microphones d'observation, permet au compère de suivre en temps réel les activités menées dans DOMUS. Le compère, installé dans la salle de la régie, utilise MOz pour remplir deux rôles :

- Simulation des capteurs : présence/absence de l'utilisateur dans une pièce, présence/absence de l'utilisateur sur le lit, utilisation du porte-serviettes, actionnement du bouton starter.
- Simulation des prédicats de la Base de Connaissances. Nous reviendrons sur ce point dans la section qui suit à propos de l'infrastructure d'accueil.

La Figure V-4 montre l'IHM du service MOz correspondant à la simulation des capteurs. Cette IHM, avec la représentation 2D du DOMUS réel, reprend celle du metteur au point mis à la disposition de l'utilisateur final (voir chapitre IV, section III-2). Rappelons que les capteurs virtuels sont vus sur le réseau comme des dispositifs UPnP au même titre que les capteurs réels. Par glisser-déposer, le compère, qui surveille les déplacements de l'utilisateur participant, place la marionnette Homer au bon endroit dans la représentation graphique du DOMUS virtuel. Par un clic souris, il désigne les représentations du porte-serviettes, du bouton starter et du lit lorsque le participant les utilise dans le DOMUS réel.

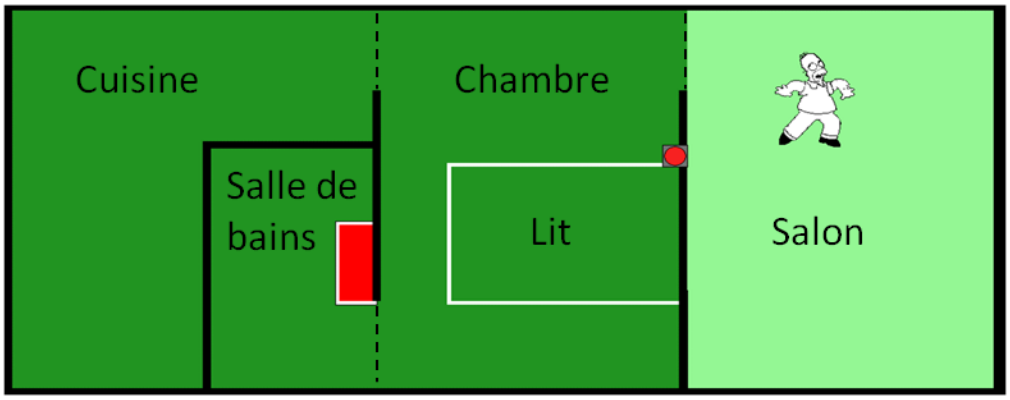


Figure V-4. IHM du service MOz utilisée par le compère pour pallier les limitations des capteurs du DOMUS réel. Le compère déplace Homer (représentant de l'utilisateur participant) dans le DOMUS virtuel de manière synchronisée avec les déplacements de l'utilisateur dans le DOMUS réel. Une zone sur fond clair indique la présence actuelle simulée. Le bouton starter est représenté par le disque rouge à côté du lit ; le porte-serviettes de la salle de bain, par le rectangle rouge. Le compère désigne ces représentations par un clic souris lorsque le participant les utilise dans le DOMUS réel.

[42] MOz est réalisé avec la boîte à outils COMETs (Demeure et al. 2008).

Nous venons de présenter les adaptations guidées par l'extrait de scénario pour, *in fine*, disposer d'une séquence réaliste et réalisable dans DOMUS. Nous considérons maintenant les adaptations techniques relatives à l'infrastructure d'accueil.

I-2 Influence de l'infrastructure d'accueil

KISS, on l'a vu, repose sur l'infrastructure CONTINUUM et son intergiciel WCOMP (cf. chapitre IV, section I). Nous détaillons dans cette section comment nous avons concilié les capacités techniques de DOMUS et de CONTINUUM/WCOMP pour répondre à nos besoins pratiques d'intégration logicielle ainsi qu'à nos requis de latence et de robustesse. Ces adaptations se répartissent en quatre points : extension logicielle pour les équipements UPnP, remplacement de WCOMP, simulation de la Base de Connaissances et développement d'un second appartement virtuel.

I-2-1 Extension logicielle des équipements UPnP

Dans WCOMP, on le rappelle, tout équipement UPnP est représenté, dans le container applicatif, par un composant proxy (voir section I du chapitre IV). À chaque apparition ou disparition de proxy, l'AA Designer de WCOMP identifie les points de coupe pour ensuite reconfigurer les composants du container applicatif. La recherche des points de coupe s'opère sur les métadonnées des équipements (type de l'équipement, localisation, taille, etc.) pour lesquelles le protocole UPnP n'offre pas de standard de représentation.

En conséquence, WCOMP introduit son standard en imposant l'existence d'un service « métadonnées ». Nous avons dû implémenter ce service pour tous les équipements UPnP de DOMUS. Ce travail a été relativement aisé pour les équipements dont la partie UPnP avait été développée par l'équipe MultiCom du LIG (équipe qui gère DOMUS). Pour les équipements dont nous n'avions pas le code source (cas de l'audio notamment), nous avons dû développer des proxy enrichis du service « métadonnées ».

I-2-2 Remplacement de WCOMP par un intergiciel dédié

Les phrases de l'éditeur KISS sont traduites en Aspects d'Assemblages (AA) (voir section IV-5 du chapitre IV). Si des tests concluants ont été réalisés en laboratoire sur un petit nombre de composants et d'AAs, il n'en a pas été de même avec le passage à l'échelle dans DOMUS. En effet, en tant que développeurs de KISS, nous avions besoin de vérifier la correction des AAs générés par KISS (par exemple, les liaisons entre les composants). L'environnement WCOMP offre un outil de mise au point, mais comme le montre la figure V-5, son interface graphique est quelque peu inadaptée à la représentation de nombreux composants.

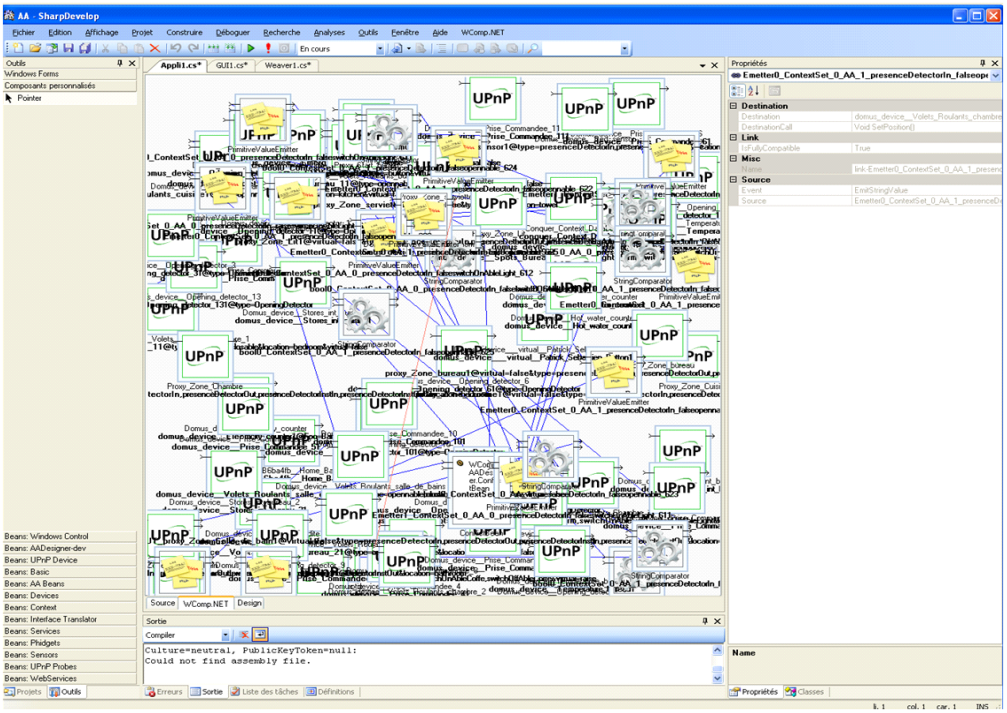


Figure V-5. Etat de l'IHM de l'outil de mise au point de WCOMP après le tissage de composants correspondant à la phrase « Quand je rentre dans la chambre, fermer tous les volets et allumer la lumière de la chambre avec une intensité moyenne ».

Outre la difficulté de la mise au point, plusieurs problèmes sont apparus lors des pré-tests dans DOMUS :

- Détection incomplète des appareils UPnP lorsqu'ils existent en grand nombre (de l'ordre de la cinquantaine) et notamment des lumières indispensables au scénario expérimental. Etait-ce dû à une défaillance de la pile UPnP WCOMP utilisée ?

- Instabilité de WCOMP allant de l'affichage en boucle de fenêtres d'erreur (peut être dû à des AA mal définis) à des crashs de l'intergiciel. Il nous a été difficile, voire impossible, d'identifier l'origine de ces erreurs et donc d'y remédier.

Ces difficultés nous ont poussés à implémenter un intergiciel de remplacement dit "de secours" avec la boîte à outils COMETs (Demeure et al. 2008). Cet intergiciel de secours, développé sur mesure en 330 lignes de Tcl, ne vise pas à concurrencer WCOMP. Il permet seulement de :

- observer les composants UPnP présents sur le réseau et détecter ceux qui sont dotés du service « métadonnées ».

- gérer des listes d'AA. Comme dans WCOMP, ces AA ont une partie condition (qui porte sur la présence de composants) et une partie action (qui relie des composants).

L'intégration de l'intergiciel de secours a été facilitée par l'architecture de KISS dans laquelle les briques dépendantes de l'infrastructure d'accueil sont clairement identifiées. Ainsi, seul le générateur de langage cible a nécessité une réécriture

(Section III-5 du chapitre IV) pour remplacer les AA WCOMP par des AA « maison » compris par l'intergiciel de secours.

I-2-3 Simulation de la Base de Connaissances (BdC)

Une première version de la BdC a fait l'objet d'une intégration avec KISS, mais nous n'avons pas pu intégrer la dernière version de la BdC dont le développement technique évoluait constamment. Au vu des besoins de notre expérimentation, une simulation simplifiée de la BdC s'est révélée être un bon compromis.

En effet, notre extrait de scénario utilise un nombre limité de prédicats : d'une part, la séquence est courte et se déroule à un instant précis de la journée limitant *de facto* le nombre de situations et de contextes ; d'autre part, les prédicats que la BdC doit traiter ne portent que sur des conditions d'aspect duratif. Or, dans le scénario expérimental, la majorité des conditions est d'aspect inchoatif.

La BdC simulée (figure V-6) constitue la deuxième fonction de MOz. L'IHM qui lui correspond présente son état sous la forme d'une liste de couples <case à cocher – prédicat>. Le compère coche ou décoche les cases en fonction des situations observées. Lorsque le Gestionnaire de Contexte (GdC) envoie une demande d'évaluation d'un prédicat donné pour la première fois, ce prédicat est ajouté à la liste et la BdC lui renvoie *Faux*. Aux demandes suivantes d'évaluation de ce même prédicat, la BdC retourne *Vrai* si la case est cochée, *Faux* dans le cas contraire.

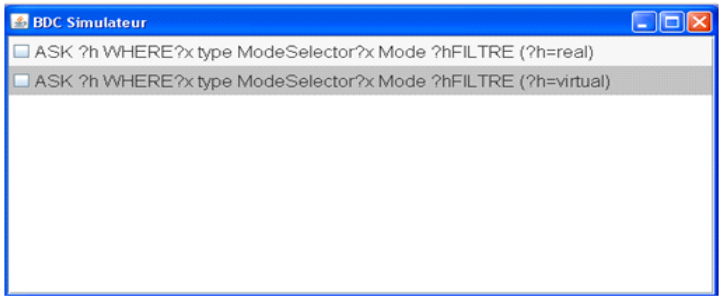


Figure V-6. IHM du service MOz utilisée par le compère pour simuler l'évaluation de prédicats. Chaque ligne est un prédicat dont l'évaluation est demandée par le GdC. La réponse retournée par MOz dépend de l'état de la case à cocher correspondante. Ici deux prédicats, l'un qui exprime que les AA à déployer correspondent à ceux du monde réel, l'autre à ceux du monde simulé. Le déploiement peut se faire dans les deux à la fois.

I-2-4 L'habitat virtuel et le metteur au point

Nous rappelons que le metteur au point mis à la disposition de l'utilisateur final est un double virtuel de l'habitat réel (voir Chapitre IV, section III-2-2). Un habitat virtuel 3D a d'abord été développé à l'aide du moteur de jeu Blender par un stagiaire de l'équipe IIHM. Les tests préliminaires dans DOMUS de ce metteur au point ont montré que la pile UPnP implémentée dans Blender ne supportait pas la charge : de nombreux appareils virtuels n'étaient pas "vus" sur le réseau, d'autres ne répondaient pas aux commandes. Le développement de l'appartement 3D ayant été réalisé par un stagiaire qui, depuis, avait quitté l'équipe, nous avons estimé que le temps de prise en main et de correction du code était trop important. Nous avons donc opté pour le

développement, avec la boîte à outils COMETs, d'un équivalent 2D dont la pile UPnP répond aux requis de latence et de fiabilité.

La figure V-7 représente l'interface de l'appartement simulé déjà présenté au chapitre précédent. Pour rappel, nous trouvons : HOMER, l'avatar de l'utilisateur que l'utilisateur final peut déplacer par glisser-déposer ; les différentes pièces de DOMUS (cuisine, chambre, salle de bains, salon) ; les équipements : le lit dans la chambre, les volets (en bas de l'interface) dont la couleur peut être noire, grise ou blanche selon leur état (fermés, entr'ouverts ou ouverts) ; les lumières représentées par des points noirs quand elles sont éteintes, et jaune quand elles sont allumées ; le bouton starter (près du lit) ; les haut-parleurs dans la chambre ; le porte-serviettes et l'indicateur de température dans la salle de bain ; la cafetière dans la cuisine.

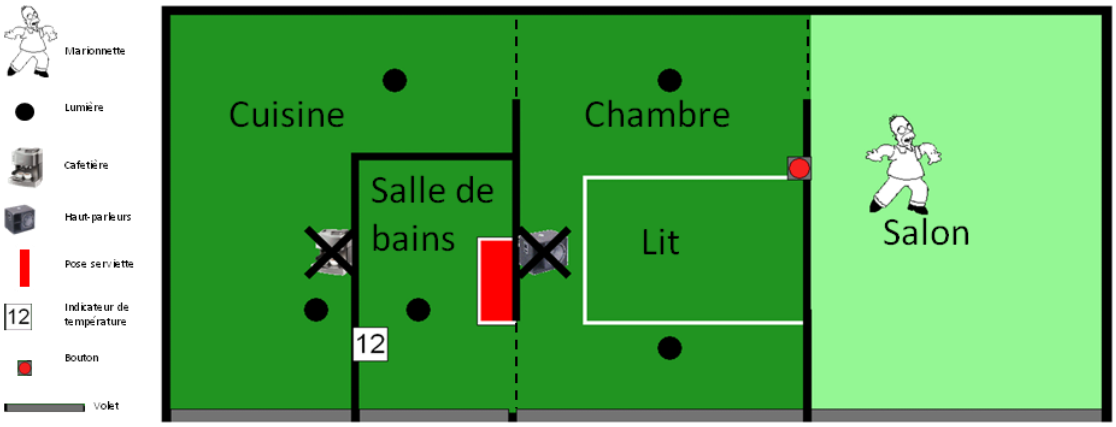


Figure V-7. L'IHM de l'appartement virtuel 2D utilisée pour la mise au point des phrases rédigées par l'utilisateur.

En synthèse, le transfert des travaux de laboratoire vers une structure d'accueil de type intelligence ambiante comme DOMUS exige des adaptations inattendues qu'il faut savoir prévoir dans un plan de développement. Nous tirons de cette expérience des enseignements et des recommandations que nous synthétiserons dans notre chapitre de conclusion.

Ces différents ajustements aboutissent à un extrait de scénario programmable par l'utilisateur final et applicable dans DOMUS. On trouvera en annexe V-3 l'interface graphique du Magicien d'Oz ainsi que KISS tel qu'il est présenté aux participants pour les tests d'acceptabilité. La section suivante concerne le protocole expérimental retenu.

II – Protocole expérimental

L'objectif du test d'acceptabilité est double :

- D'une part, valider l'usage du langage pseudonaturel comme moyen de programmer le comportement d'un habitat intelligent. En particulier, il s'agit de vérifier que l'utilisateur peut exprimer aisément ses objectifs avec le

vocabulaire fourni, qu'il comprend quels sont les objets sur lesquels porte le programme qu'il rédige et qu'il maîtrise la sémantique de ses programmes.

- D'autre part, valider l'utilité d'une réplique virtuelle de l'habitat réel pour la mise au point des programmes. En particulier, il s'agit de vérifier que l'utilisateur est capable de se repérer dans l'habitat virtuel et d'y retrouver les objets qu'il programme.

Nous synthétisons le déroulement du protocole expérimental, puis nous présentons le recrutement des participants. Le déroulement complet du test peut être consulté en annexe V-1.

II-1 Déroulement de l'expérimentation

Le protocole expérimental comprend six étapes : accueil, visite de l'appartement, formation à l'utilisation de KISS, exécution du scénario par le participant, vérification du bon déroulement du scénario, débriefing avec questionnaire.

1) Le participant est accueilli sur le site par l'expérimentateur qui lui présente l'expérimentation en quelques mots. A cette étape, il lui est demandé de signer un consentement pour l'enregistrement des données expérimentales.

2) L'expérimentateur lui fait visiter l'appartement DOMUS. Il lui fournit un plan de DOMUS (celui de la figure V-2) et lui montre pièce par pièce les différents équipements, y compris les capteurs. Une fois la visite terminée, l'expérimentateur s'assure que le participant a compris l'agencement de DOMUS et de ses équipements.

3) Le participant est formé à l'utilisation de KISS (éditeur de phrases, appartement virtuel et horloge). Pour cela, l'expérimentateur rédige une phrase en langage pseudonaturel (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 au participant 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 UPnP et de lui en expliquer le fonctionnement. Enfin, l'expérimentateur présente l'appartement simulé. Il explique qu'il s'agit d'une reproduction de DOMUS et y situe

les pièces et équipements. L'expérimentateur propose au participant de déplacer l'avatar dans la salle de bain, de le sortir, puis de refaire la même manipulation en changeant l'heure. À la fin de cette étape, l'expérimentateur s'assure que le participant a bien compris le fonctionnement des outils qui lui sont fournis.

4) Le participant doit programmer la séquence *du réveil matinal* dans DOMUS à l'aide des outils qu'il vient de prendre en main (voir encadré de la figure V-8). 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 l'éditeur de phrases. 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 posées par le participant. 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 aller le tester dans l'appartement accompagné de l'expérimentateur.

Vous habitez dans cette maison intelligente qui contient des objets programmables : machine à café, volets, lumière, capteurs de présence, porte-serviettes, etc. Grâce à ces nouveaux dispositifs, vous allez pouvoir programmer votre maison et en particulier l'enchaînement des tâches du matin.

Sur la table de chevet, un bouton vous permet de lancer la routine du matin. 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.

Tout n'est pas forcément envisageable ?

Nous vous demandons de faire cette programmation et d'expliquer à haute voix ce que vous faites et pourquoi. Vous pouvez utiliser le langage naturel, le simulateur et l'horloge virtuelle. Quand vous pensez avoir fini, dites-le nous. Si vous n'y arrivez pas, ne vous inquiétez pas : le problème vient certainement de DOMUS.

Figure V-8. Le scénario du réveil matinal soumis aux participants pour programmer DOMUS avec KISS.

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 de nouveau.

6) Le participant est convié à un débriefing au cours duquel il peut faire 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 libérer le participant, il lui est demandé de remplir un questionnaire d'évaluation (Annexe V-2).

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

II-2 Recrutement des participants

Nous avons effectué notre test d'acceptabilité avec 13 participants recrutés par courriel et par relation personnelle.

Notre échantillon comprend six femmes et sept hommes. Comme nous souhaitions avoir de « véritables » utilisateurs finaux, nous avons limité à 4 le nombre de professionnels informaticiens. Parmi les neuf autres, cinq sont des cadres supérieurs, et quatre ont une profession intermédiaire. Tous nos participants ont une source de revenu. La moyenne d'âge de notre échantillon est de 39 ans. La tranche d'âge est comprise entre 26 et 66 ans en suivant la répartition donnée dans le tableau V-1. Ce tableau répertorie également les catégories socioprofessionnelles des participants.

Tableau V-8. Répartition des âges dans l'échantillon de participants.

Sujets	Tranche d'âges	Professions et Catégories Sociales
S1	26 à 30 ans	Professions Intermédiaires
S2	40 à 66 ans	Cadres supérieurs
S3	26 à 30 ans	Professions Intermédiaires - Informaticien
S4	26 à 30 ans	Professions Intermédiaires - Informaticien
S5	40 à 66 ans	Cadres supérieurs - Retraité
S6	30 à 40 ans	Professions Intermédiaires
S7	30 à 40 ans	Professions Intermédiaires - Informaticien
S8	40 à 66 ans	Cadres supérieurs - Retraité
S9	26 à 30 ans	Professions Intermédiaires - Informaticien
S10	30 à 40 ans	Professions Intermédiaires
S11	40 à 66 ans	Professions Intermédiaires
S12	30 à 40 ans	Employés-Ouvriers
S13	30 à 40 ans	Professions Intermédiaires

Le corpus de données comprend 26 heures d'enregistrement vidéo et audio accompagnées d'une grille d'analyse des questionnaires et d'une analyse thématique.

III - Résultats du test d'acceptabilité

Dans cette section, nous commençons par discuter de la façon dont le scénario a été perçu par les participants avant de présenter les résultats des tests vis-à-vis de notre approche en général et de KISS en particulier.

D'une manière générale, à l'exception d'un participant (S1), le scénario a été compris par les participants. S1 a éprouvé des difficultés à la lecture du scénario, il réfléchissait à l'implication de toutes les phrases entre elles et n'arrivait pas à le décomposer. Il a fallu une intervention rassurante de notre part pour que S1 puisse démarrer le test. D'une manière moins bloquante, la décomposition du scénario a posé quelques problèmes à S2. Ce dernier avait du mal à ordonner les phrases par rapport à la chronologie du scénario. Dans le même ordre d'idée, le préchauffage de l'eau a perturbé plusieurs participants (S4, S5, S13) : ils ignoraient dans quelle phrase placer cette action de sorte qu'elle soit chronologiquement valide par rapport au scénario. Après discussion, il est apparu que la sémantique du terme "préchauffer" était ambiguë pour ces participants contrairement au terme "chauffer".

Une des causes du stress initial du participant S1 venait de l'usage du bouton starter dans le scénario. Dans une moindre mesure, nous avons constaté que d'autres participants ont également été perturbés par ce bouton. La majorité des participants n'a pas saisi l'intérêt de ce bouton. « *Le rôle du bouton n'est pas clair* » (S2), « *On peut enlever le bouton, il ne sert à rien, car quand je sors du lit c'est suffisant* » (S8), « *Le bouton, ne sert à rien* » (S9), « *le bouton, je m'en moque* » (S11). Nous avions inclus ce bouton en référence aux résultats de notre étude de terrain (cf chapitre II). Il devait servir au participant à gérer les cas d'exceptions : une séquence d'actions usuelles peut exceptionnellement ne pas être déclanchée si l'utilisateur ne le souhaite pas. Seuls quelques participants semblent avoir perçu l'importance de l'exception à la routine dans ce scénario mais pour aucun d'entre eux le bouton n'est apparu comme une bonne solution.

Le porte-serviettes est également apparu comme source de difficultés (S12, S7). Sa relation avec le déclenchement de la cafetière n'a pas été évidente pour S1.

La cafetière a incité les participants à s'approprier le scénario ce qui a parfois créé quelques difficultés. Par exemple, S2 ne comprend pas l'intérêt d'allumer la cafetière puisque la sienne est toujours sous tension. S6 a quant à lui cherché à piloter la gazinière car c'est comme ça qu'il fait son café.

Dans la suite de ce chapitre, nous présentons les résultats du test d'acceptabilité en deux parties. La première partie porte sur l'appréciation générale de notre approche, à savoir, la programmation de l'habitat à l'aide d'un langage pseudonaturel et d'un habitat virtuel. La seconde partie porte sur l'appréciation plus particulière de KISS en tant qu'outil. Chacune de ces parties est divisée en trois sections : les aspects positifs retenus par les participants, une description des difficultés rencontrées suivie d'une discussion sur les pistes d'amélioration.

III-1 Résultats relatifs à notre approche

III-1-1 Aspects positifs

Les participants ont globalement apprécié l'expérience. Ils ont trouvé l'interaction avec l'habitat « *amusante* » (S3, S6), « *sympathique* » (S9, S12), « *ludique* » (S13). Cette

appréciation générale nous rassure sur la volonté des utilisateurs finaux à contrôler leurs habitats intelligents.

Les participants sont d'accord (10/13) pour indiquer que l'utilisation de l'éditeur de phrases et du simulateur est un bon moyen pour programmer l'appartement. 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 :

Ceux qui ont besoin du langage pseudonaturel et de l'habitat virtuel,

Ceux qui n'ont besoin que du langage

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.

Ces résultats sont cohérents avec les fréquences d'utilisation des outils : quatre participants aimeraient utiliser fréquemment l'habitat virtuel (S2, S4, S5, S12) parmi lesquels nous retrouvons trois des quatre participants (S2, S5, S12) à avoir exprimé le besoin d'utiliser le LPN ainsi que l'habitat virtuel pour programmer DOMUS.

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 paraît 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. Nous n'avons pas évalué ce point plus avant bien qu'il soit possible que S8 ait raison.

Tous les participants sont d'accord pour affirmer 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 onze 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 KISS (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.

III-1-2 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. Il semblerait donc que le langage naturel ne soit pas la meilleure modalité à employer par un utilisateur final pour la spécification de contraintes horaires. Ce résultat est à rapprocher d'une étude très ancienne de C. Plaisant et de B. Shneiderman sur ce sujet précis (Plaisant et al. 1991).

Si pour l'ensemble des participants (13/13), le LPN est cohérent (« *compréhensible, pas ambiguë* » (S3, S4)), 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 que *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 « *Quand je ne suis pas dans la salle de bain, éteindre la lumière de la salle de bain* ». 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 duratif. Les participants ont toujours bien interprété le cas inchoatif « *quand je rentre* » alors qu'ils ont parfois mal interprété les cas duratif « *je suis dans* » et *« il est entre* ». Il est donc possible que l'aspect (inchoatif/duratif) de la condition ait un impact sur l'interprétation qui en est faite par les participants.

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). Enfin, pour (S5), le LPN *« manque de personnalisation, trop mécanique, manque d'humanité* ».

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 des difficultés rencontrées, certains participants ont eu le sentiment d'avoir commis des erreurs : « *L'appartement a compris, c'est moi qui ai eu 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, malheureusement* » (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), « *A 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 pour (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 élevé : « *Dans une grosse maison, ça deviendrait compliqué* » (S5).

Besoin de service à valeur ajoutée

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 une valeur ajoutée.

Les participants de notre test d'acceptabilité ont manifesté l'envie d'acquérir un tel système, mais pas sous n'importe quelle condition. Trois participants ont précisé qu'un « *système facilitant c'est bien, mais de temps à autre, il peut être agréable de pouvoir reprendre la main* » (S5, S9), « *Ça, c'est fait pour faciliter la vie, mais j'aime bien maîtriser ce que je fais. Le dimanche, j'ouvre les volets moi-même* » (S10). Ces remarques confirment la nécessité de disposer de points de contrôles ouverts sur l'utilisateur tel que définis dans l'introduction de cette thèse ainsi que dans le Chapitre IV avec le concept de méta-IHM.

Absence de prise en compte de plusieurs habitants au sein du foyer

Enfin, quelques sujets se sont inquiétés de l'usage d'un tel environnement de développement dans un habitat intelligent habité par une famille. Les questionnements portaient sur le lien entre les programmes des différentes personnes (S8) et en particulier comment les conflits entre des règles contradictoires pouvaient être résolus (S5). Ces inquiétudes touchent l'une des limitations de ce travail de thèse, à savoir que nous nous sommes limités à l'étude de la programmation de l'habitat par un habitant unique.

III-1-3 Pistes d'améliorations

Personnalisation et extension du vocabulaire

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 de saisie des phrases (S1 et S2). Une piste envisageable est l'usage d'un vocabulaire co-construit (Barraquand 2012) où l'utilisateur et le système apprennent l'un de l'autre comme dans TellMe (Gil et al. 2011). TellMe essaie d'associer les termes qu'il ne comprend pas à des concepts qu'il connaît en questionnant périodiquement l'utilisateur, ceci au risque d'être disruptif.

Dans le même ordre d'idées, 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* ». Si KISS inclut un éditeur de grammaire (voir chapitre IV, section IV-2), il n'est pas, en l'état, destiné à un utilisateur final.

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 l'exemple* » (Girard 2000). 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.

Ordonner/grouper les phrases

Certains participants ont pensé 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 les étapes entre elles* » (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 amont de terrain (Coutaz et al.

2010). De plus, plusieurs participants ont émis l'idée que des groupes de phrases pourraient avoir la priorité sur 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* » comme 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 au confort. « *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 l'importance de l'exception à la routine. Nous avions proposé dans le scénario de gérer cette exception via l'usage du bouton. Comme mentionné ci-dessus, le bouton a posé plusieurs problèmes. S5 estime que le bouton est inutile puisque, pour lui, la discrimination entre la routine et l'exception peut être détectée par la différence de chemin prise par l'utilisateur : si l'utilisateur n'effectue pas les étapes habituelles, le système doit comprendre que le scénario à employer est celui de base.

Utilisation du multisyntaxe

Le LPN s'est trouvé parfois inadapté, et notamment pour l'expression de contraintes horaires. L'approche multisyntaxe avec égale opportunité totale est clairement souhaitable. KIxsSS, qui intègre un langage abstrait pivot, ouvre cette possibilité, mais seule la version LPN est actuellement opérationnelle.

III-2 Résultats relatifs à KISS

III-2-1 Aspects positifs

Les participants sont d'accord pour affirmer que l'habitat virtuel qui sert de metteur au point est accessible (11/13). Il ne demande pas trop de formation (9/13). Cependant, trois sujets, non-informaticiens, déclarent avoir besoin d'une assistance technique pour l'utiliser (S10, S12, S13). De plus, S10 a clairement exprimé que l'habitat virtuel ne lui « *parle pas* » et qu'il a « *besoin de concret* ». Un seul sujet l'a trouvé inutilement complexe et estime que « *dans un système fiable, cet outil ne sert à rien* » (S8).

De manière informelle, nous avons demandé aux participants leurs impressions sur l'habitat virtuel 3D que nous avions prévu à l'origine. Celui-ci a été jugé plus réaliste que l'habitat 2D mais « *lourd* » (S9), « *trop compliqué* » (S7), « *moins pratique que le plan 2D* » (S12). « *Le 3D est un jeu ; là, on veut être efficace ; le plan grossier 2D est génial* » (S2). Un sujet seulement semble préférer le 3D : *« les gens commencent à être habitués au 3D, avec les visites de musée* »(S5). De manière générale, les participants

partagent l'avis que le plan 2D est mieux adapté que l'interface 3D pour tester un programme rapidement et efficacement.

L'éditeur supporte bien la construction des phrases. Il a été jugé « *assez précis, facilement manipulable* » (S9), et « *relativement intuitif* » (S11) avec cependant des limites que nous résumons ci-dessous.

III-2-2 Difficultés rencontrées

À l'évidence, l'éditeur de phrase appelle des améliorations : « *Son ergonomie est à revoir* » (S5, S6), «*la manipulation via les menus est fastidieuse* »(S3) et la réutilisation de phrases n'est pas possible (S9). Un participant a souhaité que la programmation se fasse vocalement : « *j'aurais préféré du parler et de la magie, mais il faut aussi de l'écriture* » (S2).

Plusieurs participants ont éprouvé des difficultés à spécifier le lieu d'une action. En effet, ils avaient tendance à manipuler l'avatar du simulateur en espérant ainsi contraindre la localisation des actions mentionnées dans la phrase en cours de rédaction. Par exemple, déplacer l'avatar dans la chambre pour signifier qu'il faut "allumer les lumières" de la chambre. Pour d'autres, il paraît évident que le lieu spécifié dans la partie condition contraint celui de l'action à être le même : « *si je suis dans la pièce X, l'action se passera dans la pièce X* » (S2, rejoint par S13).

Lors des tests avec le metteur au point, certains participants n'arrivaient pas à comprendre « *quelle phrase était en train d'être exécutée* » (S2). En d'autres termes, ils n'arrivaient pas à établir une correspondance entre les phrases rédigées et les effets (ou l'absence d'effets) observés dans l'habitat virtuel. De même, « *quand quelque chose se passe mal, il n'est pas évident de savoir si le bug vient du système ou si c'est notre phrase qui est mal rédigée*» (S4). C'est là un manque de l'éditeur qui aurait pu mettre en exergue les phrases au moment de leur exécution.

Pour finir, alors que tous les sujets ont confiance en DOMUS et en ses capteurs (13/13), il est intéressant de noter que ce n'est pas le cas lorsqu'il s'agit de l'habitat virtuel (8/13). (S9) et (S12) ne partagent cependant pas cet avis : « *le virtuel montre bien que le réel va fonctionne*r » (S12), « *en virtuel, la machine à café, ça fonctionne. Pas besoin de voir en vrai* » (S9). On peut aussi observer que les participants semblent être moins confiants en leur programmation qu'en DOMUS, « *DOMUS a compris ce que je lui ai dit, c'est moi qui me suis trompé* » (S13).

III-2-3 Pistes d'améliorations

Guidage pour la rédaction des phrases

L'éditeur LPN permet actuellement de rédiger des phrases paradoxales comme : « *allumer la lumière de la chambre et éteindre la lumière de la chambre* » (S4), « *quand il est 7h et quand il est 8h* » (S1), « *quand je suis dans la salle de bains et quand je suis dans la cuisine* » (S2). Une première amélioration consisterait à ne proposer que des choix cohérents. « *Quand je suis dans la salle de bains, je voudrais avoir les actions en*

lien avec la salle de bain » (S2) ; « *quand je paramètre la chambre, j'allume les lumières de la chambre* » (S13).

D'autres formes de guidage ont été suggérées par les participants comme le fait que le système puisse faire des rappels du type « *le lieu de l'action n'est pas spécifié* » (S5), « *vous n'avez pas précisé d'heure* » (S10). D'autres participants aimeraient que, lors du survol d'un élément de menu, son correspondant dans l'habitat virtuel soit mis en évidence. Ceci permettrait découvrir plus facilement « *ce qui se cache derrière les termes de l'éditeur* » (S3, S4, S5).

Un participant a proposé l'usage de cases à cocher, à la façon des formulaires, à l'ouverture d'un menu relatif au lieu (S4). Ainsi, il serait possible de sélectionner un ensemble de pièces où une action aura lieu, par exemple : « *ouvrir les volets de la chambre, la cuisine, mais pas ceux de la salle de bain* ».

Langage dédié à la définition de contraintes horaires

La pose des contraintes horaires a été difficile pour plusieurs participants. L'un d'eux a proposé de modifier le texte "il est", présent dans le premier menu des conditions, par une icône représentant une horloge (S12). De cette façon, l'utilisateur doit comprendre immédiatement que cela porte sur un horaire. Un autre participant propose d'utiliser un tableur avec des horaires, « *à la façon d'un doodle* ». Cette piste n'est pas à écarter : l'utilisation du temps est un problème qui mérite des recherches plus approfondies.

Facilité de réutilisation des phrases

Dans la version actuelle de l'éditeur, le système ne permet que la modification d'une phrase existante. Cependant, de nombreux participants ont exprimé le besoin de réutiliser des phrases déjà rédigées pour en créer de nouvelles. Cette réutilisation leur apporterait un gain de temps : ils pourraient par exemple réutiliser une phrase en n'en modifiant que le lieu d'exécution. Plusieurs participants ont généralisé cela en suggérant l'utilisation de phrases "patrons", prêtes à paramétrer (S1, S2, S5, S8).

Il a aussi été proposé que l'habitat soit "livré" avec un ensemble de phrases « *par défaut* » (S7) encodant « *des choses basiques* » (S5). Cet ensemble de phrases serait modifiable et extensible par les utilisateurs : « *allumer la lumière de la pièce quand j'y suis, c'est de base, après je mets une heure* » (S2). Ces propositions ont souvent été émises en réaction à la perception que le système complexifiait des choses qui auraient dû être simples (ex : allumer les lumières d'une pièce si l'utilisateur s'y trouve) : « *le système complexifie des actions déjà prises en compte de manière simple et automatique* » (S4).

Unification des outils

L'unification des trois outils proposés en un seul a été proposée comme moyen de limiter le nombre d'équipements pour programmer l'habitat (S2, S5, S7, S9). Les participants ont généralement imaginé qu'une tablette pourrait être suffisante pour programmer et tester (S2, S5, S7).

Un autre argument pour l'unification des outils est de faciliter la génération d'illustrations dans l'habitat virtuel des phrases spécifiées par les utilisateurs (S1, S7, S11, S12, S13). Les utilisateurs voudraient que le système fasse « *la démonstration de ce que l'on a fait* » (S12) et propose la possibilité de « *tester phrase par phrase* » (S11), de « *jouer la phrase qui vient d'être programmée via un bouton play* » (S7). Ce comportement souhaité correspond à la définition de la programmation avec exemples de Girard (Girard 2000).

La programmation sur l'exemple a également été énoncée comme piste d'amélioration. Il s'agirait de manipuler l'habitat virtuel pour programmer (S1, S2, S5, S7, S11) : « *avec des menus textuels sur les objets du monde virtuel* » (S1) où l'utilisateur « *joue dans le virtuel et clique-droit pour savoir quoi faire* » (S7). Cette programmation sur l'exemple est envisagée comme moyen de contraindre la construction des phrases : « *placer Homer [l'avatar] dans le lit pour contraindre la localisation des phrases au lit* » (S5), « *programmer grossièrement avec le virtuel, puis au besoin, affiner avec les phrases* » (S5). On retrouve le principe d'égale opportunité.

IV – Conclusion

L'évaluation expérimentale montre que les participants ont pu programmer avec succès le scénario imposé. L'utilisation du langage pseudo-naturel et l'utilité d'un habitat virtuel dual de l'habitat réel ont été bien comprises. Les difficultés rencontrées portent pour l'essentiel sur le vocabulaire limité et les quiproquos quant à l'interprétation de certaines phrases incluant des verbes duratifs. 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 sur et avec exemples. D'autres pistes d'améliorations ont été proposées, et dans l'ensemble, ces résultats nous confortent dans la validation des objectifs de cette thèse tout en nous ouvrant des voies pour de futurs travaux.

Chapitre VI : Conclusion

Table des matières :

I - L'apport de la thèse 157
II – Du laboratoire au terrain : nos recommandations..... 162
III – Considérations éthiques..... 164

Cette thèse traite de la construction d'espaces intelligents par l'utilisateur final. Elle part du constat que dans la pratique actuelle des processus de développement, l'utilisateur est réduit à consommer des systèmes conçus et produits par des spécialistes qui ne peuvent répondre, malgré les méthodes éprouvées, à des besoins évolutifs et mal définis. En outre, ces spécialistes, considérant l'autonomie comme « la » qualité d'un système ambiant, sortent l'utilisateur de la boucle et de là, lui retirent le nécessaire contrôle.

L'objectif de cette thèse est de redonner le pouvoir à l'utilisateur par le biais d'outils de « Développement par l'Utilisateur Final » (DUF). Cet objectif se fonde sur l'hypothèse que l'utilisateur est un créateur d'espaces intelligents constructibles par assemblage incrémental de dispositifs et de services. Par conséquent, ce travail de recherche doctoral a consisté à : (1) vérifier la validité de notre hypothèse, (2) développer un outil DUF en sorte que l'utilisateur puisse construire à façon et contrôler facilement et de manière opportuniste, des espaces intelligents, et (3) valider l'acceptabilité de cet outil par l'expérimentation avec des utilisateurs représentatifs. Par pragmatisme, nous nous sommes limités à une classe particulière d'espaces intelligents : l'habitat des familles en relation avec les activités du quotidien.

Nous souhaitons ici conclure ce mémoire en trois points : les éléments clefs de l'apport de cette thèse, les leçons qu'il convient de tirer de la confrontation de solutions conçues en laboratoire avec les exigences expérimentales du terrain, et une question ouverte sur l'éthique.

I - L'apport de la thèse

L'apport de cette thèse comprend trois volets : méthode, taxonomie et réalisation technique. Il s'agit de :

(1) *DisQo (Dispositifs du QuOtidien),* une nouvelle méthode d'investigation pour faire entrevoir les services domestiques du futur.

(2) Un *espace de classification* pour une lecture comparative systématique et synthétique des outils portant sur le développement et la programmation d'habitats intelligents.

(3) *KISS (Knit Your Ideas into Smart Spaces),* un outil de programmation et de mise au point centré sur l'habitat intelligent jouant le rôle de méta-IHM dans une infrastructure d'exécution pour systèmes ambiants.

I-1 DisQo : une méthode pour faire entrevoir les services domestiques du futur

Nous recommandons, avant d'appliquer une méthode comme DisQo dont la visée est ciblée, de conduire une investigation auprès du monde économique de façon à

identifier à gros-grains les tendances du marché. C'est ce que nous avons fait sous forme d'entretiens auprès de professionnels de l'habitat et du particulier. Cette enquête préliminaire nous a permis de confirmer, dès la phase amont, qu'effectivement, il existe une demande de la part des particuliers pour équiper leur habitat d'objets « intelligents ». Elle confirme également que le coût d'acquisition de ces équipements et la difficulté de les programmer constituent les principaux freins à leur déploiement.

DisQo répond à des exigences quelque peu incompatibles :

- respecter la sphère privée des foyers et limiter le temps de présence de l'équipe d'investigation, mais recueillir des données écologiquement valides ;
- faire entrevoir et imaginer des services futurs alors que l'intelligence ambiante est à peine naissante, voire inexistante, dans les foyers.

À ces fins, DisQo combine plusieurs outils de mesure et notamment des entretiens semi-directifs *in situ* et une sonde culturelle ludique. L'élément clef de DisQo est la possibilité donnée aux observateurs de visiter l'habitat à travers les photos prises par les participants, puis d'utiliser ces photos d'objets personnels comme sonde culturelle ludique. Cette utilisation de photos présente de multiples avantages :

- elle établit une relation de confiance entre les familles et les observateurs ;
- elle suscite l'intérêt et la curiosité des familles (« *que va-t-il se passer ensuite ?* ») ;
- elle permet de partager avec les familles des avis sur la nature des objets pris en photo et de là, les classer comme éléments structurants d'émergence de sens (indispensables ou à forte valeur intime, communicants et/ou programmables) ;
- les familles révèlent leur habitat et leurs habitudes tout en préservant leur espace privé ;
- l'utilisation d'objets personnels pour le jeu des associations sollicite l'imagination mais, s'agissant d'objets familiers, limite la charge cognitive.

L'analyse des données recueillies selon la méthode DisQo confirme des résultats connus de la littérature : l'existence de moments clefs journaliers marqués par des activités routinières qui ne souffrent pas l'exception, et des classes de besoins comme le confort matériel, monitoring, aide-mémoire, aide à la décision, gestion du temps et des stocks. Mais, de manière plus originale, DisQo nous apprend que, dans un habitat intelligent :

- la capacité, pour un objet, d'être communicant influe plus fortement sur l'émergence de sens que sa capacité d'être programmable ;
- quatre types de couplage de services sont attendus : (1) *Substitution* de service c'est-à-dire la possibilité de remplacer, dans un assemblage préexistant, un service par un autre de meilleure qualité (ou perçu comme tel). Par exemple, spécifier l'heure d'alarme du radio-réveil physique via l'IHM de l'horloge d'un

SmartPhone. (2) *Augmentation* d'objets du quotidien, c.-à-d. améliorer un objet grâce aux ressources d'un autre. Par exemple, utiliser la TV comme moyen de contrôle d'une lave-linge qui, lui, ne dispose pas d'écran. (3) *Chaînage* de services, c.-à-d. mettre des services bout à bout avec enchaînement automatisé afin de gagner du temps dans l'accomplissement de tâches routinières. (4) *Déclenchement manuel*, c.-à-d. disposer d'un déclencheur pour contrôler les automatismes.

- Les couplages de services sont largement contraints par des relations temporelles avec une présence marquée pour les opérateurs d'Allen meet (« *La machine à café démarre quand je finis ma douche* »), start with (« *La musique commence à me suivre dès que j'allume la lumière* ») et during (« *Tout en écoutant la musique, je me fais aider par mon armoire pour choisir mes vêtements de la journée* »).

En synthèse, DisQo constitue une nouvelle méthode qui sollicite l'imagination des participants pour aboutir à un équilibre satisfaisant entre contrôle expérimental, respect de la sphère privée et validité écologique, le tout en 1h30 environ par séance expérimentale. Il convient cependant de compléter DisQo par un journal de bord de manière à récolter des idées après la séance.

I-2 Un espace de classification des outils PUF-DUF-GLUF pour habitats intelligents

La littérature est riche en outils PUF-DUF-GLUF, mais peu a été fait sur les espaces de classification et encore moins concernant l'habitat intelligent. Un espace de classification permet de raisonner sur l'existant de manière structurée et systématique : analyse et comparaison de solutions, identification des tendances et des manques, et de là, création de nouvelles solutions.

Concernant les classifications en matière de PUF-DUF-GLUF, deux articles font référence : (1) « Lowering the barriers to programming : a Survey of Programming Environments and languages for novice programmers » (Kellher et al. 2003) et (2) « The State of the art in End-user Software Engineering » (Ko et al. 2011). Le premier introduit une distinction intéressante entre les « outils pour apprendre à programmer » et les « outils qui donnent du pouvoir à l'utilisateur ». Le second va plus loin que le PUF-DUF, mais l'état de l'art y est structuré par la seule couverture des activités de développement et le support à la qualité logicielle. Aucun des deux espaces n'entrevoit la fusion des mondes réels et virtuels ni celle des phases de conception et d'exécution comme des éléments distinctifs. Rien n'est dit sur le multisyntaxe et l'égale opportunité dont on a constaté l'importance dans notre expérimentation de terrain.

L'espace de classification proposé dans cette thèse reprend les aspects clefs des deux contributions précédentes (et d'autres, plus classiques) avec ses propres extensions. Il permet de répondre à trois grandes classes de questions sur :

1. Le rôle de l'utilisateur final : est-il un consommateur, un délégateur, un constructeur ?
2. Les propriétés de l'environnement de développement et d'exécution : quelle en est la couverture fonctionnelle ? Existe-il un support au développement collectif ? Quel couplage entre phase de développement et phase d'exécution ?
3. Les propriétés du langage de programmation : est-il extensible ? Est-il générique ou spécifique à un domaine auquel cas, existe-t-il une correspondance directe entre les unités du langage et les concepts et tâches du domaine, de même existe-t-il une structuration en niveaux de complexité croissante ? De quel paradigme relève-t-il ? Quels sont les facteurs favorisant la lisibilité et la facilité d'écriture des programmes ? En particulier, quel est le support à la correction lexicale, syntaxique et sémantique mais aussi, existe-t-il un support multisyntaxe avec égale opportunité couplé à plusieurs modalités de spécification ?

Le placement, dans notre espace de classification, des outils PUF-DUF-GLUF pour l'habitat intelligent, montre que :

- les forces actuelles de ces outils portent essentiellement sur la spécificité, l'extensibilité et le support à la correction ;
- une tendance nette se dégage pour le paradigme orienté règles couplé à des syntaxes dites de « langage visuel » et de modalités de spécification WIMP ;
- les faiblesses essentielles concernent la dichotomie entre phase de développement et phase d'exécution, la réutilisation, et trop peu d'égard pour l'expression des relations temporelles et le multisyntaxe avec égale opportunité totale ;
- l'absence de considération explicite pour la dualité habitat numérique-habitat physique ou de support au co-développement.

Ces résultats ont servi d'éléments directeurs à la réalisation de KISS et à son intégration dans une infrastructure d'exécution pour systèmes ambiants.

I-3 KISS, un outil DUF servant de méta-IHM tactique

I-3-1 Point de contrôle ouvert sur l'utilisateur ou Méta-IHM

Nous savons aujourd'hui que le principe du « système 100% autonome » prôné par les chercheurs en systèmes répartis doit s'accommoder du principe de « l'utilisateur dans la boucle » avancé par les chercheurs en IHM. Nous proposons pour cela que tout système ambiant doit définir explicitement des points de contrôle qui « donnent la main » à l'utilisateur de même qu'il existe des points de reprise en système d'exploitation pour effectuer des traitements particuliers sur interruption, par exemple.

Au sein de l'infrastructure CONTINUUM, qui nous a servi de plate-forme d'intégration et d'exécution, nous avons défini trois points de contrôle ouverts sur l'utilisateur auxquels nous donnons le nom de méta-IHM (une IHM au-dessus du reste) : méta-IHM réflexe, tactique, et stratégique, par analogie avec les niveaux de compétences cognitives humaines, auxquelles sont associés des requis de latence spécifique.

Nous avons développé une méta-IHM réflexe pour en démontrer la faisabilité, mais notre effort s'est porté sur KISS en tant qu'outil DUF servant de méta-IHM tactique. L'intégration de KISS dans une infrastructure véritablement orientée intelligence ambiante et développée par des chercheurs tiers, est en soi un résultat technique original : aucun outil DUF de l'état de l'art pour habitat intelligent n'a été développé au-dessus d'une infrastructure tiers.

I-3-2 KISS

Parmi les lacunes de l'état de l'art citées ci-dessus à propos des outils PUF-DUF-GLUF, nous avons retenu celles que nous avons jugées fondamentales au regard de la vie de tous les jours et de la faisabilité dans le temps imparti à cette thèse : la dichotomie conception-exécution, la dualité habitat réel-habitat numérique, le support à la mise au point et le multisyntaxe avec égale opportunité. Quant aux relations d'Allen, nous sommes allés au plus simple, seul l'opérateur start with est supporté, pour supporter les autres opérateurs, un investissement technique est nécessaire.

KISS (Knit your Ideas Into Smart Spaces) s'adresse à un utilisateur final constructeur. Sa couverture fonctionnelle inclut la programmation, la mise au point et, dans une moindre mesure, la réutilisation de programmes. Le langage de programmation est de type déclaratif orienté règles, avec potentiellement l'égale opportunité syntaxique entre langue française pseudo-naturelle (LPN) et langage visuel iconique.

La technique d'interaction de construction des programmes LPN s'appuie sur l'utilisation de menus dont les options sont calculées dynamiquement à partir de l'état de l'habitat intelligent. KISS assure ainsi la découverte progressive du langage ainsi que l'extensibilité et la correction syntaxique et sémantique. La mise au point peut se pratiquer au choix, dans le monde physique, dans un monde dual numérique ou, dans une version ultérieure de KISS, dans les deux simultanément. Le grain de fusion des phases de conception et d'exécution est celui de la phrase.

KISS a fait l'objet d'une validation par l'expérimentation avec 13 utilisateurs en milieu semi-contrôlé. L'évaluation de KISS dans DOMUS, un habitat intelligent d'expérimentation, montre que les utilisateurs parviennent à programmer un scénario réaliste de la vie réelle. L'utilisation du langage pseudo-naturel et l'utilité d'un habitat virtuel dual de l'habitat réel ont été comprises et validées.

Les difficultés rencontrées militent pour deux améliorations essentielles que nous prenons pour des recommandations à l'adresse de la communauté scientifique : (1) pousser la personnalisation et l'extensibilité du langage par co-construction système-utilisateur (approche par apprentissage) ; (2) pousser comme nous le pensions d'emblée, l'édition multisyntaxe avec égale opportunité totale et modalités de

programmation complémentaires. En particulier, l'habitat virtuel présenté aujourd'hui comme metteur au point peut aussi servir d'outil de spécification par démonstration en différé ou en temps réel. En mode « temps réel », l'habitat virtuel devient une « télécommande » avec effet immédiat sur le monde réel. Il s'agit alors d'une méta-IHM réflexe.

Au final, l'habitat virtuel peut jouer trois rôles : metteur au point, support à la programmation par démonstration (méta-IHM tactique), télécommande de l'habitat réel (ou méta-IHM réflexe). Ce glissement de rôle rendu possible par l'architecture de CONTINUUM et de KISS, est un résultat original par rapport à l'état de l'art.

L'évaluation dans DOMUS a aussi été l'occasion de mesurer le fossé entre un contexte de laboratoire et celui du terrain.

II – Du laboratoire au terrain : nos recommandations

Notre expérience d'évaluation expérimentale appelle deux remarques essentielles. L'une concerne le respect de conditions expérimentales extrêmement exigeantes, l'autre porte sur le choix de l'infrastructure d'exécution.

II-1 À propos des conditions expérimentales

Le déploiement d'un système ambiant développé par des tiers, comme l'infrastructure CONTINUUM/WCOMP, dans un environnement inconnu de ces tiers (en l'occurrence, DOMUS), requiert de délicates adaptations. Il convient alors de savoir jouer sur les compromis de façon à :

- assurer une robustesse logicielle et matérielle suffisante pour une bonne conduite des expérimentations,
- satisfaire les requis de latence pour éviter tout risque de rejet immédiat de la part des sujets,
- respecter le réalisme et la pertinence écologique du recueil des données.

Dans notre cas, nous avons étudié le nécessaire équilibre entre de nouveaux développements logiciels ad-hoc (comme notre middleware de secours) et la révision du scénario expérimental. Grâce à l'étude amont faite avec DisQo, ce scénario était représentatif de la vie du quotidien. Il s'agissait donc de ne pas le dénaturer.

Nous retenons de cette expérience trois leçons essentielles que nous souhaitons partager avec la communauté de recherche en intelligence ambiante :

- Prévoir dans la mise en œuvre d'un prototype de système ambiant l'intégration d'un composant Magicien d'Oz de façon à remplacer par un compère humain, les entités défaillantes (ou non implémentées).

- Imposer la présence sur le terrain expérimental de toute l'équipe de développement du logiciel à tester, y compris les logiciels et matériels tiers, de façon à corriger rapidement les bugs ou les pannes. On peut ainsi s'éviter des développements ad-hoc inutiles[43].
- Une compétence scientifique unique ne suffit pas pour approcher les questions levées par la mise en œuvre de systèmes ambiants. La multidisciplinarité en un même lieu est une nécessité ou alors, gageons que les chercheurs/développeurs de systèmes ambiants se dotent d'outils collaboratifs efficaces permettant la production, coordination et communication à distance ; et cela à grande échelle. Nous en sommes loin !

II-2 À propos du choix de l'infrastructure d'exécution

Nous considérons qu'une infrastructure à composants orientés services comme CONTINUUM/WCOMP est fondamentalement un bon choix. Considérant le développement de nos méta-IHM réflexe (pour photo-browser) et méta-IHM tactique (KISS), nous obtenons gratuitement le déploiement et la reconfiguration dynamiques, l'introduction de composants d'expérimentation comme le magicien d'Oz ou des dispositifs maison.

Ces avantages ont leur contrepartie :

- *Alignement des concepts computationnels*. L'utilisateur (de l'infrastructure) doit s'adapter aux traits de l'infrastructure (conçue initialement à d'autres fins qu'au développement d'applications interactives comme KISS) et se doit de découvrir comment en contourner les limites. La traduction des programmes LPN en AAs est à cet égard révélatrice. Autrement dit, les unités conceptuelles des uns (en l'occurrence, celles de l'infrastructure) ne sont pas toujours alignées avec celles des composants applicatifs (ici, KISS).
- *Complexification*. Une infrastructure se juge par sa puissance fonctionnelle. À son tour, cette puissance se traduit par des traitements qui peuvent se révéler inutilement complexes. Par exemple, allumer une lampe, qui est une action simplissime, met en jeu dans KISS-CONTINUUM une chaîne de traitements dont la longueur peut avoir un impact sur la fiabilité. Ne pourrait-on pas faire plus simple sans perdre en puissance et généralité ?

[43] Sans la présence de l'équipe Multicom qui gère DOMUS, sans la présence de l'auteur des Comets, nous n'aurions pu effectuer l'expérimentation. Nous sommes intimement persuadés que si nous avions pu disposer de l'un des développeurs de WCOMP sur place, nous aurions réussi à utiliser WCOMP et éviter ainsi le développement du « middleware de secours ».

III – Considérations éthiques

Notre sujet mérite que nous nous attardions quelque peu sur des considérations éthiques. Parmi les nombreuses questions éthiques que soulève cette thèse, nous en retenons deux : d'une part, le bien-être et l'autonomie, la responsabilité d'autre part.

III-1 Bien-être et autonomie

Pour le bien-être et l'autonomie, toute personne est amenée à se poser des questions contradictoires exprimées en à peu près en ces termes :

- « *Pour que la programmation que j'ai faite de mon habitat intelligent améliore mon bien-être, il faut qu'il puisse identifier ma position ainsi que celle de mes enfants. Or je ne veux pas être suivi* ».
- « *Je ne veux pas être suivi, mais j'aimerais être certain que mes enfants sont à la maison. Je vais donc leur imposer quelque chose que je me refuse à faire sur moi-même. Et si eux veulent savoir que je suis à la maison ?* »

Nous estimons qu'il faut permettre à chaque individu de mener la vie autonome de son choix. Cette thèse a introduit le concept de point de contrôle ouvert sur l'utilisateur. Il s'agit là d'un élément de réponse technique, mais pas une réponse de nature morale. Faut-il donc que les machines aient un comportement moral ? « Cette question dépasse la vue classique de la perte de contrôle des machines par l'humain. Respecter des valeurs éthiques signifie que le système ambiant doit être en mesure de modéliser les valeurs de ses usagers de façon à prendre ou aider à prendre les « bonnes » décisions en cas de conflits. Actuellement, nous ne savons pas spécifier un comportement moral. L'approche simpliste consisterait à considérer les valeurs éthiques comme des contraintes supplémentaires que le système devrait satisfaire comme il en va pour toute contrainte. Encore faut-il que ces contraintes soient identifiées. Elles peuvent être très spécifiques (réduire la consommation d'énergie) ou très abstraites (ne jamais blesser un humain) » (Coutaz et al. 2008).

III-2 Responsabilité

Illustrons le problème de la responsabilité avec l'exemple suivant : un utilisateur a programmé un comportement d'habitat intelligent qu'il a rendu disponible à d'autres utilisateurs. Un second utilisateur utilise ce comportement dans son habitat intelligent. Malheureusement, un accident grave en découle. Se pose alors la question de la responsabilité. Qui est responsable ? L'utilisateur qui a programmé ? L'utilisateur qui a utilisé ? Le concepteur qui a permis à l'utilisateur de programmer ?

Nous pouvons envisager de nombreuses solutions techniques pour éviter un grand nombre de programmations à risque, mais tout comme il est impossible de supprimer les couteaux qui coupent ou les fuites de gaz, nous ne pouvons pas éviter

techniquement toutes les situations programmatiques à risque. Ou bien la société se crée des règles dont la production peut venir du législateur, du juge, des mécanismes « d'auto régulation ». La question de la responsabilité reste donc ouverte.

Ces considérations éthiques ne font qu'effleurer le sujet. Nous n'avons malheureusement pas pu approfondir ces questions. Nous estimons néanmoins qu'il est important de les poser.

Bibliographie

[Allen 1984]	Allen, J.F. "Towards a general theory of action and time" eds. Inderjeet Mani, James Pustejovsky, and Robert Gaizauskas. *Artificial Intelligence* 23(2), 1984, 123-154.
[Amabile 1983]	Amabile, T.M. "The social psychology of creativity: A componential conceptualization." *Journal of personality and social psychology* 45(2), 1983, 357-376.
[Ash et al. 2011]	Ash, J., Babes, M., Cohen, G., Jalal, S., Lichtenberg, S., Littman, M., Marivate, V., Quiza, P., Ur, B., Zhang, E. "Scratchable devices: user-friendly programming for household appliances." *Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments*, 2011, 137-146.
[Balme 2008]	Balme, L. "Interfaces homme-machine plastiques: une approche par composants dynamiques." Thèse de l'Université de Grenoble, 2008.
[Bandura 1997]	Bandura, A. "Self-efficacy: The exercise of control." *New York Freeman*, 1997.
[Barraquand 2012]	Barraquand, R. "Designing sociable technologies." Thèse de l'université de Grenoble, 2012.
[Benyelloul et al. 2010]	Benyelloul, A., Jouanot F., Rousset, M.C. "Conquer: an RDFS-based Model for Context Querying." *Proc of Ubimob 2010*, ACM, 2010, 1-4.
[Bernhaupt et al. 2007]	Bernhaupt, R., Weiss, A., Obrist, M., Tscheligi, M. "Playful Probing: Making Probing More Fun" eds. Maria Cecília Calani Baranauskas et al. *Proceedings of the 11th IFIP TC 13 international conference on Human Computer Interaction* 4662, 2007, 606-619.
[Bieber 2001]	Bieber, G., Carpenter, J. "Introduction to Service-Oriented Programming" *OpenWings Whitepaper April*, 2001, 1-13.
[Bigot et al. 2008]	Bigot, R., Croutte, P. "La diffusion des technologies de l'information et de la communication dans la société française." *CREDOC*, 2008.
[Blackwell et al. 2002]	Blackwell, A.F., Rob, H. "AutoHAN: An architecture for programming the home." In *Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments*, 2002, 150–157.
[Blackwell et al. 2002]	Blackwell, A.F., Burnett, M. "Applying attention investment to end-user programming." *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments* (1), 2002, 28-30.
[Brandt et al. 2008]	Brandt, J., Guo, P.J., Lewenstein, J., Klemmer, S.R. "Opportunistic Programming: How Rapid Ideation and Prototyping Occur in Practice." *In WEUSE '08: Proc. 4th Workshop on End-User Software Engineering*, ACM, 2008, 1-5.
[Buechley et al. 2008]	Buechley, L., Eisenberg, M., Catchen, J. "The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education." *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, 2008, 423-432.
[Burnett 1999]	Burnett, M. "Visual Programming." *Wiley Encyclopedia of Electrical and Electronics Engineering*, 1999.
[Calvary 2003]	Calvary, G. "A Unifying Reference Framework for multi-target user interfaces." *Interacting with Computers* 15(3), 2003, 289-308.
[Calvary 2007]	Calvary, G. "Plasticité des Interfaces Homme-Machine.", Thèse Habilitation à Diriger des Recherches, Université de Grenoble, 2007.

[Carroll 2000]	Carroll, J.M. “Making use: scenario-based design of human-computer interactions. ” *MIT Press Cambridge*, 2000.
[Carroll et al. 1984]	Carroll, J. M., Carrithers C. “Training wheels in a user interface.” *Communications of the ACM*, 27(8), 1984, 800-806.
[Cheung 2009]	Cheung-Foo-Wo, D. “Adaptation dynamique par tissage d’aspects d'assemblage.” Thèse de l’université de Nice-Sophia Antipolis, 2009.
[Cockton 2004]	Cockton, G. “From quality in use to value in the world.” *Extended abstracts of the 2004 conference on Human factors and computing systems - CHI ’04*, 2004, 1287-1290.
[Cockton 2005]	Cockton, G. “A development framework for value-centred design.” In *CHI ’05 extended abstracts on Human factors in computing systems - CHI '05*, New York, ACM Press, 2005, 1292-1295.
[Combessie 2007]	Combessie, J.C. “La méthode en sociologie”, collection Repères. *Edition la découverte*, 2007.
[Conway et al. 1997]	Conway, M.J., Miksad, W. “Alice: Easy-to-Learn 3D Scripting for Novices.” *University of Virginia*, 1997.
[Coutaz et al. 2005]	Coutaz, J., Crowley J., Dobson, S., Garlan, D. “Context is key” *Communications of the ACM* 48(3), 2005, 49-53.
[Coutaz 2007]	Coutaz, J. “Meta-user interfaces for ambient spaces.” *Task Models and Diagrams for Users Interface Design* 9, 2007, 1–15.
[Coutaz et al. 2008]	Coutaz, J., Crowley, J. “Intelligence ambiante : défis et opportunités”. Rapport Document de réflexion conjoint du comité d’experts « Informatique Ambiante » du département ST2I du CNRS et du Groupe de Travail « Intelligence Ambiante » du Groupe de Concertation Sectoriel (GCS3) du Ministère de l’Enseignement Supérieur et de la Recherche, DGRI A3, 2008.
[Coutaz et al. 2010]	Coutaz, J., Fontaine, E., Mandran, N., Demeure, A. “DisQo : A user needs analysis method for smart home.” *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, 2010, 615-618.
[Coutaz et al. 2011]	Coutaz, J., Calvary, G. “HCI and Software Engineering for User Interface Plasticity”. *In The Human Computer Interaction Handbook - Fundamentals, Evolving Technologies, and Emerging Applications*, 3rd Edition, Chapter 52, Taylor & Francis Publ, 2011.
[Coutaz et al. 2012]	Coutaz, J., Crowley, J. “Intelligence Ambiante : effet de mode ou discipline d'avenir ? ” Dans *Informatique et Intelligence Ambiante : des capteurs aux applications*, 2012.
[Coutaz et al. 2012]	Coutaz, J., Calvary, G. “Systèmes interactifs et adaptation centrée utilisateur : la plasticité des Interfaces Homme-Machine”. *Dans Informatique et Intelligence Ambiante : des capteurs aux applications*, Chapitre 9. Hermes Publ., 2012.
[Dahlback et al. 1993]	Dahlback, N., Jonsson, A., Ahrenberg L. “Wizard of Oz studies — why and how.” *Knowledge-Based Systems* 6(4), 1993, 258-266.
[Davidoff et al. 2006]	Davidoff, S., Lee, M., Yiu, C., Zimmerman, J., Dey, A.K. “Principles of smart home control.” *UbiComp 2006*, 2006, 19–34.
[Davidoff et al. 2012]	Davidoff, S., Dey, A.K., Zimmerman, J. “Adding place and time to family calendars. (in submission).” 2012.
[Demeure et al. 2008]	Demeure, A., Calvary, G., Coninx, K. “COMET(s), A Software Architecture Style and an Interactors Toolkit for Plastic User Interfaces.” *Interactive Systems Design Specification and Verification*, 2008, 225-237.
[Dey et al. 2004]	Dey, A.K., Hamid, R., Beckmann, C., Li, I., Hsu, D. “a CAPpella: programming by demonstration of context-aware applications.” In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 2004, 33-40.

[Dey et al. 2006]	Dey, A.K., Sohn, T., Streng, S., Kodama, J. "iCAP: Interactive prototyping of context-aware applications." *Pervasive Computing*, 2006, 254–271.
[Dix 2002]	Dix, A. "Beyond intention-pushing boundaries with incidental interaction." In *Proceedings of Building Bridges: Interdisciplinary Context-Sensitive Computing, Glasgow University*, 2002.
[Drey 2010]	Drey, Z. Recherche "Vers une méthodologie dédié a l'orchestration d'entitées communicantes.", thèse de l'université de Bordeaux 1, 2010.
[Edwards et al. 2001]	Edwards, W., Grinter, R. "At home with ubiquitous computing: Seven challenges." In *Ubicomp 2001: Ubiquitous Computing*, Springer, 2001, 256–272.
[Edwards et al. 2001]	Edwards, W.K., Newman M.W., Sedivy J.Z. "The case for recombinant computing." *Xerox Palo Alto Research Center Technical Report CSL-01-1*, 2001, 1-19.
[Ellis et al. 1991]	Ellis, C.A., Gibbs S.J., Rein G. "Groupware: some issues and experiences." *Communications of the ACM* 34(1), 1991, 39–58.
[Escoffier et al. 2007]	Escoffier, C., Hall, R., Lalanda, P. "iPOJO: an Extensible Service-Oriented Component Framework." In *Services Computing 2007 SCC 2007 IEEE International Conference on*, IEEE Computer Society, 2007, 474-481.
[Ferry et al. 2010]	Ferry, N., Lavirotte, S., Tigli, J.Y., Rey, G., Riveill, M. "Toward a Behavioral Decomposition for Context-awareness and Continuity of Services." In *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence (ISAmI 2010)*, 2010, 55-62.
[Ferry 2011]	Ferry, N. "Adaptations dynamiques au contexte en informatique ambiante: propriétés logiques et temporelles." Thèse de l'Université de Nice-Sophia Antipolis. 2011.
[Ferry et al. 2011]	Ferry, N., Hourdin, V., Lavirotte, S., Rey, G., Riveill M., Tigli, J.Y. "WComp, a Middleware for Ubiquitous Computing." In *Ubiquitous Computing*, InTech, 2011, 151-176.
[Fisher et al. 2006]	Fisher, M., Rothermel, G., Brown, D., Cao, M., Cook, C., Burnett, M. "Integrating automated test generation into the WYSIWYT spreadsheet testing methodology." *ACM Trans Softw Eng Methodol* 15(2), 2006, 150-194.
[Fogg 2003]	Fogg, B.J. "Persuasive Technology Using Computers to Change What We Think and Do." *Morgan Kaufmann*, 2003.
[Fontaine et al. 2012]	Fontaine, E., Demeure, A., Coutaz, J., Mandran, N. " Programmation d'environnements intelligent par des utilisateurs finaux, un retour d'expérience" (in submission), 2012.
[Gabillon 2011]	Gabillon, Y. "Composition d'Interfaces Homme-Machine par planification automatique." Thèse de l'Université de Grenoble, 2011.
[Gantt et al. 1992]	Gantt, M., Nardi B.N. "Gardeners and gurus: patterns of cooperation among CAD users." In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, eds. Penny Bauersfeld, John Bennett, and Gene Lynch. ACM, 1992, 107-117.
[Gaver et al. 1999]	Gaver, B., Dunne, T., Pacenti E. "Cultural Probes." *interactions* 6(february), 1999, 21-29.
[Gallissot 2012]	Gallissot, M. "Modéliser le concept de confort au sein de l'habitat intelligent : du multisensoriel au comportement." Thèse de l'Université de Grenoble. 2012.
[Gil et al. 2011]	Gil, Y., Ratnakar, V., Frtiz C. "Tellme: Learning procedures from tutorial instruction." In *Proceedings of the 15th international conference on Intelligent user interfaces*, ACM, 2011, 227–236.

[Girard 2000]	Girard, P. “Ingénierie des systèmes interactifs: vers des méthodes formelles intégrant l’utilisateur.” Thèse pour l’obtention de l’habilitation à diriger les recherches de *LISI/ENSMA*, 2000.
[Gorbet et al. 1998]	Gorbet, M.G., Orth, M., Ishii, H. “Triangles: tangible interface for manipulation and exploration of digital information topography.” In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press/Addison-Wesley Publishing Co., 1998, 49–56.
[Grudin 1994]	Grudin, J. “CSCW: History and Focus.” *IEEE Computer*, 1994, 19–26.
[Guilford 1987]	Guilford, J.P. “Creativity research: Past, present and future.” *Frontiers of creativity research: Beyond the basics*, 1987, 33–65.
[Hague et al. 2006]	Hague, R., Robinson, P. “End-user programming of reconfigurable systems.” *Software: Practice and Experience* 36(11-12), 2006, 1285-1306.
[Hague et al. 1997]	Hague, R., Robinson, P. “Multi-Lingual End-User Programming with XML.” *Lingua*, 1997, 34-40.
[Harrison et al. 1996]	Harrison, S., Dourish P. “Re-place-ing space: the roles of place and space in collaborative systems.” In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, ACM, 1996, 67–76.
[Harrison et al. 2010]	Harrison, C., Tan, D., Morris, D. “Skinput: appropriating the body as an input surface.” In *Proceedings of the 28th international conference on Human factors in computing systems*, ACM, 2010, 453–462.
[Harrison et al. 2011]	Harrison, C., Benko, H., Wilson, A.D. “OmniTouch: wearable multitouch interaction everywhere.” In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM, 2011, 441–450.
[Hindus et al. 2001]	Hindus, D., Mainwaring, S.D., Leduc, N., Hagström, A.E., Bayley, O. “Casablanca: designing social communication devices for the home” In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 2001, 325–332.
[Hourdin et al. 2008]	Hourdin, V., Tigli, J.Y., Lavirotte, S., Rey, G., Riveill, M. “SLCA, composite services for ubiquitous computing.” *Proceedings of the International Conference on Mobile Technology Applications and Systems Mobility* 08 35(6), 2008.
[Humble et al. 2003]	Humble, J., Crabtree, A., Hemmings, T., Åkesson, K.P., Koleva, B., Rodden, T., Hansson, P. “‘Playing with the Bits’ User-configuration of Ubiquitous Domestic Environments.” In *UbiComp 2003: Ubiquitous Computing*, Springer, 2003, 256–263.
[Jacob et al. 2002]	Jacob, R.J.K., Ishii, H., Pangaro, G., Patten, J. “A tangible interface for organizing information using a grid.” *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world changing ourselves CHI 02* (1), 2002, 339-346.
[Jacob et al. 2008]	Jacob, R.J.K., Ishii, H., Pangaro, G., Patten, J. “Reality-based interaction: a framework for post-WIMP interfaces.” In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM, 2008, 201–210.
[Jang et al. 2005]	Jang, S., Ko, E.J., Woo, W. “Unified user-centric context: Who, where, when, what, how and why.” *Personalized Context Modeling and Management for UbiComp Applications* 149, 2005, 26-34.
[Jouve et al. 2008]	Jouve, W., Lancia, J., Palix, N., Consel, C., Lawall, J. “High-level Programming Support for Robust Pervasive Computing Applications.” *6th IEEE International Conference on Pervasive Computing and Communications PerCom (2008)*, 2008, 252-255.

[Kierkegaard 2011]	Kierkegaard, P. “Beefing Up End User Development: Legal Protection and Regulatory Compliance.” *End-User Development*, 2011, 203-217.
[Ko et al. 2004]	Ko, A.J., Myers B.A. “Designing the whyline: a debugging interface for asking questions about program behavior.” In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004, 151-158.
[Ko et al. 2004]	Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B. “The State of the Art in End-User Software Engineering.” *Journal ACM Computing Surveys*, 2011.
[Leshed et al. 2008]	Leshed, G., Haber, E., Matthews, T., and Lau, T., “CoScripter: Automating & sharing how-to knowledge in the enterprise.” *In Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, 2008, 1719-1728.
[Letondal 2006]	Letondal, C. “Participatory Programming : Developing programmable bioinformatics tools for end-users.” *End User Development*, 2006, 207–242.
[Lewis et al. 1987]	Lewis, C., Olson, G. “Can principles of cognition lower the barriers to programming?” *Empirical studies of programmers second workshop* 2, 1987, 248-263.
[Lieberman et al. 2006]	Lieberman, H., Paterno, F., Klann, M., Wulf, V. “End-user development: An emerging paradigm*.” End User Development*, 2006.
[Lin et al. 2008]	Lin, J., Wong, J., Nichols, J., Cypher, A., Lau, T.A. “End-user programming of mashups with vegemite.” *Proceedingsc of the 13th international conference on Intelligent user interfaces - IUI ’09*, 2009, 97-106.
[Maloney et al. 2010]	Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. “The Scratch Programming Language and Environment.” *ACM Transactions on Computing Education 10(4)*, 2010, 1-15.
[Martin 1982]	Martin, J. “Application Development Without Programmers.” *Prentice Hall PTR*, 1982.
[Mattern 2002]	Mattern, F. “Ubiquitous & Pervasive Computing: A Technology-driven Motivation.” *Summer school on ubiquitous and pervasive computing*, 2002.
[Mednick 1962]	Mednick, S.A. “The associative basis of the creative process.” *Psychological review* 69(3), 1962, 220-32.
[Mistry 2009]	Mistry, P. “SixthSense: a wearable gestural interface.” *ACM SIGGRAPH ASIA 2009 Sketches*, 2009.
[Myers 1986]	Myers, B. “What are visual programming, programming by example, and program visualization?” In *Proceedings on Graphics Interface ’86/Vision Interface '86*, 1986, 62-65.
[Myers et al. 2000]	Myers, B., Hudson S.E., Pausch, R. “Past, present, and future of user interface software tools.” *ACM Transactions on Computer-Human Interaction* 7(1), 2000, 3-28.
[Nardi 1993]	Nardi, B.A. “*A Small Matter of Programming: Perspectives on End User Computing*”. MIT Press, 1993.
[Newman et al. 2008]	Newman, M., Elliott, M., Smith, T. “Providing an integrated user experience of networked media, devices, and services through end-user composition.” *Pervasive Computing*, 2008, 213–227.
[Nigay 1994]	Nigay, L. “Conception et modélisation logicielles des systèmes interactifs: application aux interfaces multimodales.” Thèse de l’Université de Grenoble 1,1994.
[Normand 1992]	Normand, V. “Le modèle SIROCO: de la spécification conceptuelle des interfaces utilisateur à leur réalisation.” Thèse de l’Université de Grenoble 1,1992.

[Paillé et al. 2011]	Paillé, P., Mucchielli, A. “Analyse thématique”, dans *L’analyse qualitative en sciences humaines et sociales*. A. Colin, 2011.
[Pattis 1981]	Pattis, R.E. “Karel the robot: a gentle introduction to the art of programming.” John Wiley & Sons, Inc, 1981.
[Pausch et al. 2003]	Pausch, R., Kelleher, C. “Lowering the Barriers to Programming: A Survey of Programming Environments and Languages for Novice Programmers.” *Human Factors*, 2003.
[Pfaff 1985]	Pfaff, G.E. “User interface management systems.” Springer-Verlag New York, Inc. 1985.
[Plaisant et al. 1991]	Plaisant, C., Shneiderman, B. “Scheduling on-off home control devices.” *Proceedings of the SIGCHI conference on Human factors in computing systems Reaching through technology - CHI ’91*, 1991, 459-460.
[Preece et al. 2002]	Preece, J., Rogers, Y., Sharp, H. “Interaction Design: Beyond Human-Computer Interaction.” ed. Wiley Sons, 2002.
[Rasmussen 1986]	Rasmussen, J. “Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering.” *NorthHolland Series in System Science and Engineering ed. Andrew P Sage*, Elsevier Science Inc. 1986.
[Rekimoto et al. 2001]	Rekimoto, J., Ullmer, B., Oba, H. “DataTiles: a modular platform for mixed physical and graphical interactions.” In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 2001, 269-276.
[Repenning 1993]	Repenning, A. “Agentsheets: applying grid-based spatial reasoning to human-computer interaction.” *Proceedings 1993 IEEE Symposium on Visual Language*, 1993, 77-82.
[Repenning et al. 2011]	Repenning, A., Ahmadi, N., Repenning, N. “Collective Programming: Making End-User Programming (More) Social.” *End-User Development*, 2011, 325-330.
[Resnick et al. 2005]	Resnick, M., Myers, B., Nakakoji, K., Shneiderman, B., Pausch, R., Selker, T., Eisenberg, M. “Design Principles for Tools to Support Creative Thinking.” *Science* 20(2), 2005, 25-35.
[Resnick et al. 2009]	Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. “Scratch: programming for all.” *Communications of the ACM 52(11)*, 2009, 60–67.
[Rey 2005]	Rey, G. “Contexte en interaction homme-machine: le contexteur.” Thèse de l’Université de Grenoble, 2005.
[Rodden et al. 2004]	Rodden, T., Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Akesson, K.P., Hansson, P. “Configuring the ubiquitous home.” *Cooperative systems design: scenario-based design of collaborative systems*, 2004, 227-242.
[Rode et al. 2004]	Rode, J.A., Toye E.F., Blackwell A.F. “The fuzzy felt ethnography-understanding the programming patterns of domestic appliances.” *Personal and Ubiquitous Computing* 8(3-4), 2004, 161-176.
[Rode et al. 2005]	Rode, J.A., Toye E.F., Blackwell A.F. “The domestic economy: a broader unit of analysis for end user programming.” In *CHI’05 extended abstracts on Human factors in computing systems*, ACM, 2005, 1757–1760.
[Salber 1995]	Salber, D. “De l’interaction homme-machine individuelle aux systèmes multi utilisateurs. L'exemple de la communication homme-homme médiatisée.” Thèse de l’Université Grenoble 1,1995.
[Salber et al. 1991]	Salber, D., Dey, A.K., Abowd, G.D. “The context toolkit: aiding the development of context-enabled applications.” In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit*, ACM, 1999, 434-441.

[Sale 1979]	Sale, A. “Pascal stylistics and reserved words.” *Software Practice and Experience* 9(10), 1979, 821-825.
[Scaffidi 2009]	Scaffidi, C. “Topes: Enabling End-User Programmers to Validate and Reformat Data.” Thèse de University of Nebraska-Lincoln, 2009.
[Silverman 1997]	Silverman, D. “Introducing Qualitative research.” Sage Publications Ltd. 1997.
[Simsarian 2003]	Simsarian, K.T. “Take it to the Next Stage : The Roles of Role Playing in the Design Process.” *New Horizons* 1, 2003, 1012-1013.
[Sousa et al. 2011]	Sousa, J.P., Keathley, D., Le, M., Pham, L., Ryan, D., Rohira, S., Tryon, S., Williamson, S. “TeC: End-User development of software systems for smart spaces”. In *International Journal of Space-Based and Situated Computing (IJSBSC)*, 2011, 257-269.
[Szyperski 2002]	Szyperski, C., Gruntz, D., Murer, S. “Component Software: Beyond Object-Oriented Programming”. *Addison-Wesley Professional*, 2002.
[Tennant et al. 1983]	Tennant, H.R., Ross, K.M., Thompson, C.W. “Usable natural language interfaces through menu-based natural language understanding.” In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM, 1983, 154–160.
[Tigli et al. 2009]	Tigli, J.Y., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E., Riveill, M. “WComp middleware for ubiquitous computing: Aspects and composite event-based Web services.” *Annals of Telecommunications Annales Des Télécommunications* 64(3-4), 2009, 197-214.
[Tigli et al. 2011]	Tigli, J.Y., Lavirotte, S., Rey, G., Hourdin, V., Riveill, M. “Lightweight Service Oriented Architecture for Pervasive Computing.” *Journal of Computer Science* 4(1), 2011, 1-9.
[Truong et al. 2004]	Truong, K.N., Abowd, G.D. “Inca: A software infrastructure to facilitate the construction and evolution of ubiquitous capture & access applications.” *Pervasive Computing*, 2004, 140–157.
[Truong et al. 2004]	Truong, K.N., Huang, E.M., Abowd, G.D. “CAMP: A magnetic poetry interface for end-user programming of capture applications for the home.” *UbiComp 2004: Ubiquitous Computing*, 2004, 143–160.
[Weiser 1991]	Weiser, M. “The computer for the 21st Century.” *IEEE Pervasive Computing* 265(1), 1991, 94–104.

Annexes

Table des matières :

ANNEXE II-1 : Extrait du rapport du CREDOC 174
ANNEXE II-2 : Grille d'entretien auprès des industriels 182
ANNEXE II-3 : Scénario directeur pour le développement de KISS 186
ANNEXE IV-1 : Grammaire de KISS 191
ANNEXE V-1 : Déroulement de l'expérimentation 195
ANNEXE V-2 : Questionnaire d'évaluation 199
ANNEXE V-3 : KISS pour les participants et le magicien d'Oz 201

ANNEXE II-1 : Extrait du rapport du CREDOC

Contexte sociétal par rapport à l'équipement des ménages en termes de nouvelles technologies et de mobilité.

Cette annexe est un extrait du rapport « **La diffusion des technologies de l'information et de la communication dans la société française (Novembre 2008). » Régis Bigot et Patricia Croutte.**

Enquête et rapport du CREDOC : Cet extrait présente les résultats des questions relatives à la diffusion, à l'usage et à l'acceptabilité des nouvelles technologies en France. Ces questions ont été insérées par le CGTI (conseil général des technologies de l'information) par l'ARCEP (l'Autorité de Régulation des Communications Electroniques et des Postes) dans la vague de juin 2008 de l'enquête du CREDOC sur les « Conditions de vie et les Aspirations des Français ».

Document : http://www.arcep.fr/uploads/tx_gspublication/etude-credoc-2008-101208.pdf

Les résultats présentés sont issus de l'enquête de juin 2008 réalisée auprès de deux échantillons distincts, représentatifs, sélectionnés selon la méthode des quotas : le premier porte sur 2011 personnes de 18 ans et plus, le second sur 210 individus âgés de 12 à 17 ans. Toutes les interviews se sont déroulées « en face à face », à domicile.

Sommaire

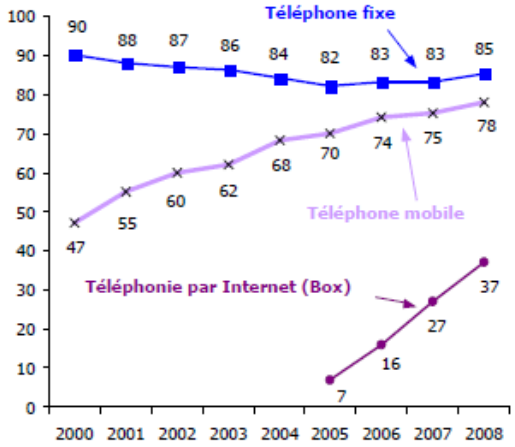
- La téléphonie par Internet, en se développant très rapidement, relance l'équipement en téléphone fixe 175
- L'Internet et la télévision sur le téléphone mobile peinent à séduire le grand public 175
- Plus de deux personnes sur trois sont équipées d'un ordinateur et, neuf fois sur dix, elles disposent d'un accès à Internet 26
- Pas d'engouement pour les connexions à Internet en mobilité 176
- Internet occupe une place croissante dans la vie des Français 26
- Internet et environnement 176
- Inquiétudes sur la protection des données personnelles 177
- Taux d'équipement en ordinateur à domicile : 69% 177
- Internet, Taux d'équipement à domicile : 58% 177
- Lieux de connexions 177
- Connexions en mobilité 178
- Rapprochement entre les différents équipements 178
- Quelques usages du micro-ordinateur et d'Internet 180
- Les freins au développement d'Internet 181

La téléphonie par Internet, en se développant très rapidement, relance l'équipement en téléphone fixe

Il s'agit là d'un retournement : le taux d'équipement en téléphone fixe s'oriente à nouveau à la hausse, tiré par le succès des boîtiers multi-services (« triple play »), qui permettent à la fois de se connecter à Internet, de recevoir la télévision et de téléphoner. La téléphonie par Internet concerne donc aujourd'hui 37% des adultes (et 56% des adolescents).

Taux d'équipement en téléphone mobile, téléphone fixe et téléphonie par Internet

- Champ : individus de 18 ans et plus -

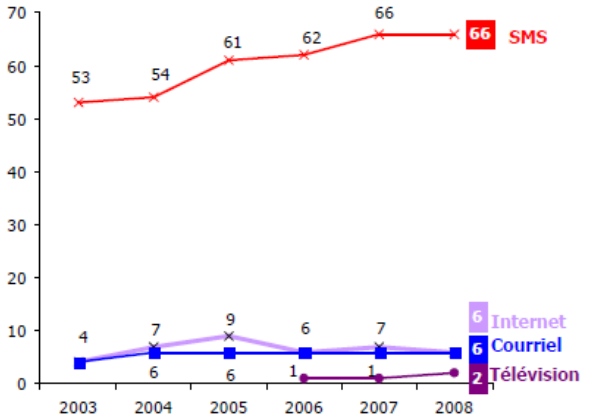


Source : CREDOC, enquêtes « Conditions de vie et Aspirations des Français ».

L'Internet et la télévision sur le téléphone mobile peinent à séduire le grand public

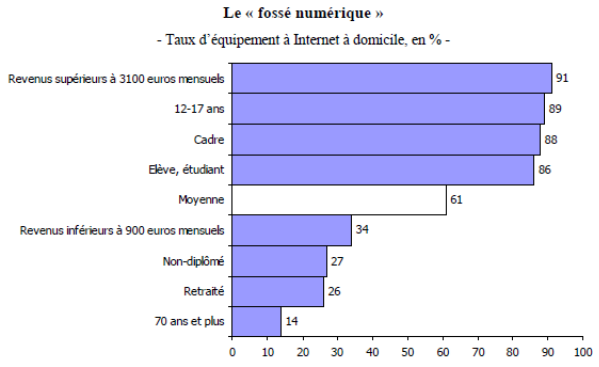
Nous suivons ces indicateurs depuis maintenant 5 ans : nous n'observons toujours aucun décollage de l'accès à Internet *via* le téléphone portable. Consulter ses mails, regarder la télévision et surfer sur la Toile à partir de son téléphone mobile restent des pratiques très concentrées dans d'étroites franges de la population dans lesquelles on retrouve quelques passionnés des nouvelles technologies et des cadres supérieurs « nomades » : **6% seulement des possesseurs de téléphone mobile l'utilisent pour naviguer sur Internet ; une proportion équivalente y consulte ses courriels et 2% y regardent la télévision.**

- Champ : individus de 18 ans et plus disposant d'un téléphone mobile -



Plus de deux personnes sur trois sont équipées d'un ordinateur et, neuf fois sur dix, elles disposent d'un accès à Internet

69% des Français disposent d'un ordinateur à domicile et 61% ont accès chez eux à Internet.



Source : CREDOC, enquête « Conditions de vie et Aspirations des Français », juin 2008.

Pas d'engouement pour les connexions à Internet en mobilité

L'accès à Internet dans les lieux publics grâce à un ordinateur portable (Wi-fi ou via un abonnement mobile notamment) reste à un niveau faible : 6% de la population en 2008, soit 1 point de plus qu'en 2007 et 2 de mieux qu'en 2006. Cette progression est très limitée, comparée avec le développement très rapide de l'accès à Internet à domicile et la très forte attirance des consommateurs pour des ordinateurs portables (41% des personnes équipées en informatique disposent en effet d'un ordinateur portable, contre 32% l'an dernier).

En fait, les consommateurs manifestent clairement une préférence pour les appareils nomades (les succès du téléphone mobile et des baladeurs MP3 en témoignent également), mais ils se connectent très peu à Internet en mobilité. Est-ce par manque d'intérêt ou d'utilité perçue ? Est-ce en raison d'une offre publique ou privée insuffisante, inadaptée ou peu ergonomique ? ou de prix trop élevés ?

Internet occupe une place croissante dans la vie des Français

De nombreux indicateurs révèlent l'importance que l'Internet a pris dans la vie des Français.

- Envoi de mails et de la recherche d'informations sont devenus des pratiques courantes
- 39% de la population adulte ont, ces douze derniers mois, réalisé des achats par Internet. Les achats par Internet progressent une nouvelle fois en 2008 (+ 4 points en un an)
- 40% ont effectué par ce canal des démarches administratives et fiscales. On observe un tassement des recours à l'administration électronique — confirmé par la consolidation du nombre de télédéclarants auprès des Impôts cette année (7,4 millions de déclarations de revenus par Internet en 2008, de même que 7,5 millions en 2007, alors qu'en 2006 on en comptait « seulement » 5,7)
- le téléchargement de musique et de films progresse à nouveau cette année : 20% des 18 ans et plus (et 56% des 12-17 ans)
- en progression, la création de sites personnels ou de blogs (10% des adultes, 53% des adolescents), le suivi de la télévision sur Internet (respectivement 8% et 22%) et l'utilisation du téléphone *via* son PC par Internet (9% et 7%).

Internet et environnement

En revanche, les répercussions du développement de l'informatique et d'Internet sur l'environnement est moins clairement perçu : 35% des Français estiment que ces produits représentent une menace pour l'environnement et le développement durable.

Il est vrai que le taux d'obsolescence de ces équipements est très élevé (on les remplace plus rapidement que les biens d'équipement traditionnels) et que leur multiplication dans les foyers contribue à accroître la consommation d'électricité de manière significative.

Inquiétudes sur la protection des données personnelles

Internet fait partie de la vie des Français mais, réciproquement, leur vie est de plus en plus souvent exposée sur Internet. Le développement des réseaux sociaux tels que Facebook ou Myspace, de même que la montée en puissance des blogs, forums, discussions et autres « chats » contribuent à une exhibition croissante des données personnelles sur le Réseau. Par ailleurs, le développement du cyber-commerce et de l'administration électronique multiplie les transmissions d'informations bancaires ou administratives. Malgré les efforts de sécurisation mis en avant, ce phénomène préoccupe les internautes depuis maintenant plusieurs années : **« la protection des données personnelles » est considérée depuis trois ans comme l'un des premiers freins à l'utilisation d'Internet, avant le prix, à égalité aujourd'hui avec la complexité des outils numériques.**

L'an dernier, nous avions mis en lumière la très forte hostilité des Français vis-à-vis de **l'utilisation qui pourrait être faite des traces** de leur navigation sur Internet. Le Conseil Général des Technologies de l'Information a voulu poursuivre les investigations en matière de protection de la vie privée, mais cette fois au sujet de la téléphonie mobile. On sait qu'il est aujourd'hui techniquement possible de suivre en temps réel le trajet d'une personne équipée d'un téléphone portable, car elle est, à tout moment, « géolocalisée » par les antennes relais de téléphonie mobile.

Or, 77% des personnes informées de cette possibilité souhaiteraient pouvoir interdire la transmission de cette localisation à des entreprises commerciales.

Taux d'équipement en ordinateur à domicile : 69%

Plus des deux tiers de la population (69% exactement, soit 67% des adultes et 92% des 12-17 ans,) sont désormais équipés d'au moins un ordinateur à leur domicile. Trois nouveaux points ont donc été engrangés cette année, soit une progression de 9 points sur deux ans.

On notera également que 20% de la population possèdent, à domicile, plusieurs ordinateurs.

Internet, Taux d'équipement à domicile : 58%

Internet continue sa progression dans les foyers français : 58% des personnes de 18 ans et plus sont désormais équipées (+ 5 points par rapport à l'an dernier) La progression est nette, de l'ordre de plus de 2 millions de nouveaux adeptes. Près de neuf personnes sur dix disposant d'un ordinateur sont équipés d'une connexion à Internet. La croissance enregistrée cette année pour Internet (+ 6 points) dépasse donc la hausse en équipement informatique domestique (+ 3 points).

De façon de plus en plus systématique, il est vrai, la présence au domicile d'un micro ordinateur se traduit par une connexion à Internet (dans 88% des cas, contre 83% l'an passé)

Lieux de connexions

Cyber-cafés, bibliothèques, etc.

La connexion à Internet dans un lieu public (cyber café, bibliothèque, bureau de poste ou autre …) reste peu développée : 11% des personnes interrogées se sont connectées de la sorte au cours des douze derniers mois (contre 12% un an plus tôt,). Et ce type de connexion reste

toujours utilisé de façon occasionnelle. Deux groupes se distinguent dans cette pratique : les jeunes et les diplômés. Un étudiant sur quatre se connecte de cette façon.

Les connexions Wi-Fi dans les lieux publics
La proportion d'individus qui se connectent à Internet dans les lieux publics grâce à une liaison Wi- Fi, en utilisant leur propre ordinateur portable, est plus faible encore : 6% de la population sont concernés (+ 1 point en un an, + 2 points en deux ans). Depuis que nous suivons cet indicateur, la progression est restée modeste. Le taux est certes plus élevé chez les diplômés du supérieur, mais il n'atteint guère que 14%.

Connexions en mobilité

In fine, nous avons recensé le nombre de personnes qui se connectaient à Internet « en mobilité », c'est-à-dire en-dehors de leur domicile ou de leur lieu de travail. Pour cela, nous avons pris en compte trois comportements distincts : les connexions dans les lieux publics de type cybercafés, bibliothèques…, les connexions dans les espaces publics grâce à une connexion Wi-Fi sur son propre ordinateur portable et les liaisons sur téléphone mobile.
En 2008, au total, 16% de la population naviguent sur Internet « en mobilité ». Ce chiffre n'évolue que très peu depuis 2005.

Comme l'an dernier, on constate que certains groupes sont plus coutumiers de ces modes de connexion « nomades » Citons les moins de 25 ans (36% des 18 – 24 ans se connectent ainsi à Internet), les étudiants (33%), les cadres supérieurs (27%) ou encore les habitants de Paris et de son agglomération (25%).
La connexion en mobilité ne se substitue donc pas, **elle se superpose à une pratique domestique déjà bien ancrée**. Pour preuve, 22% des personnes équipées en Internet à domicile se sont connectées en mobilité, contre 8% de celles qui ne sont pas équipées.

Rapprochement entre les différents équipements
Cette section s'intéresse aux liens qui existent entre les différents équipements issus des technologies de l'information. Le fait d'être équipé d'un ordinateur personnel est-il corrélé à une plus forte probabilité de détenir un téléphone mobile ? Dans quelle mesure le fait d'être équipé d'une connexion à Internet modifie-t-il la manière dont on reçoit les chaînes de télévision ? Pour répondre à ces interrogations, le Tableau 85 présente les liens existant entre tous les équipements possibles des particuliers. On apprend ainsi que :

- **Ne pas avoir de téléphone mobile maximise la probabilité d'avoir un téléphone fixe**, mais
diminue celle d'être équipé en nouvelles technologies (ordinateur, Internet, etc.). En effet, si 96% de ceux qui n'ont pas de téléphone mobile sont équipés en téléphone fixe, ils ont un accès réduit à l'ordinateur (39%, contre 69% en moyenne) et en connexion Internet (34%, contre 61%). De même, seuls 33% disposent d'un accès multiple à la télévision (46% en moyenne).

- **Ne pas avoir de téléphone fixe se traduit quasi systématiquement par la possession d'un téléphone mobile, mais entraîne un déficit d'équipement en ordinateur et Internet**. Ainsi, 93% de ceux qui n'ont pas de téléphone fixe ont un téléphone portable (+ 15 points par rapport à la moyenne). Mais, dans le même temps, ils disposent moins souvent d'un ordinateur qu'en moyenne (42%, - 27 points par rapport à la moyenne) et, surtout, sont très

peu connectés à Internet (15%, soit quatre fois moins qu'en moyenne). De même, un accès multiple à la télévision est plus rare (33% *vs* 46% en moyenne).

Ne pas avoir d'ordinateur à domicile est lié à un niveau moindre d'équipement en téléphonie, qu'elle soit fixe (72%, - 13 points) ou mobile (57%, - 21 points). L'accès à la télévision est multiple dans 31% seulement des cas (- 15 points par rapport à la moyenne). La connexion à Internet est, alors, quasi inexistante.

Comme ceux qui n'ont pas d'ordinateur**, les individus qui n'ont pas accès à Internet disposent moins souvent d'un téléphone** (68% ont le fixe et 62% un portable, en retrait respectivement de 13 et 16 points par rapport à la moyenne) ; ils ne sont que 22% à avoir l'usage d'un ordinateur à domicile (trois fois moins qu'en moyenne) et 34% à bénéficier d'un accès diversifié à la télévision (- 12 points par rapport à la moyenne).
Chez les personnes équipées d'un téléphone mobile, l'équipement en informatique est un peu
plus fréquent qu'en moyenne (78% contre 69%), de même que la connexion à Internet (69%, + 8 points).

Avoir **un ordinateur à sa disposition (et, *a fortiori,* en avoir plusieurs) augmente la probabilité d'avoir un téléphone portable (respectivement 87% et 89%, contre 78% en moyenne), celle d'accéder à Internet** (portée respectivement à 85% et 97%, contre 61% en moyenne). Ceux qui possèdent **plusieurs postes informatiques ont la particularité de recevoir la télévision de multiples** façons (pour 59% d'entre eux, contre 46% de l'ensemble de la population).

Les personnes connectées à Internet disposent presque toutes (96%) d'un téléphone fixe mais aussi, dans 88% des cas (+ 10 points par rapport à la moyenne), d'un téléphone portable. Le taux d'équipement en ordinateur frôle les 100%. Et, pour 54% d'entre elles, l'accès à la télévision est possible via différents canaux (+ 8 points par rapport à la moyenne).

Disposer de différents accès pour capter la télévision est synonyme d'un meilleur équipement en NTIC. 84% des ces personnes disposent en effet d'un téléphone portable (+ 6 points par rapport à la moyenne) ; 80% ont un ou plusieurs ordinateurs à domicile (+ 11 points) ; 72% sont connectées à Internet à domicile (+ 11 points).

On constate donc un effet d'entraînement très net : dès lors qu'une personne possède un produit issu des NTIC, la probabilité d'accéder aux autres s'accroît. C'est l'accès multiple à la télévision qui bénéficierait le moins de cet effet de contagion. D'ailleurs, ne bénéficier que d'un seul mode d'accès n'entraîne qu'un léger déficit sur les autres aspects (informatique, Web ou téléphonie).

Tableau 85
Coefficients de corrélation entre les différents équipements

- Champ : ensemble de la population -

	Téléphone fixe	Téléphone mobile	Un seul ordinateur à domicile	Plusieurs ordinateurs à domicile	Connexion à Internet	Connexion à haut débit	Plusieurs accès à la télévision
. Téléphone fixe		- 0,15	+ 0,10	+ 0,16	+ 0,40	+ 0,37	+ 0,11
. Téléphone mobile	- 0,15		+ 0,21	+ 0,13	+ 0,30	+ 0,29	+ 0,14
. Un seul ordinateur à domicile	+ 0,10	+ 0,21		- 0,49	+ 0,47	+ 0,43	+ 0,08
. Plusieurs ordinateurs à domicile	+ 0,16	+ 0,13	- 0,49		+ 0,36	+ 0,36	+ 0,13
. Connexion à Internet	+ 0,40	+ 0,30	+ 0,47	+ 0,36		+ 0,94	+ 0,19
. Connexion à haut débit	+ 0,37	+ 0,29	+ 0,43	+ 0,36	+ 0,94		+ 0,20
. Plusieurs accès à la télévision	+ 0,11	+ 0,14	+ 0,08	+ 0,13	+ 0,19	+ 0,20	

Source : CREDOC, Enquête « Conditions de vie et Aspirations des Français », juin 2008.

Exemple de lecture : Le coefficient de corrélation entre le téléphone fixe et le téléphone mobile est de – 0,15. Cela signifie que le fait d'être équipé d'un téléphone fixe est plutôt lié au fait de ne pas avoir de mobile. En revanche, la corrélation est positive entre la connexion Internet et l'équipement en téléphone fixe (coefficient de corrélation égal à + 0,40).

Quelques usages du micro-ordinateur et d'Internet

Communication

Depuis 2006, nous suivons l'activité qui consiste dans la création, sur Internet, de sites personnels ou de blogs. En juin 2008, 14% des Français de douze ans et plus (soit 22% des internautes, c'est-à-dire 2 points de plus que l'an dernier) se disent auteurs d'un blog ou d'un site, mettant en ligne textes, photos, musique …

Divertissement

Le téléchargement de musique concerne presque une personne sur quatre (24%, + 2 points), soit un internaute sur trois (37% exactement). Au total, douze millions et demi de personnes déclarent télécharger de la musique sur le Net. La pratique se maintient donc à un niveau relativement élevé. **Regarder la télévision sur Internet** est une pratique qui se banalise (10%, + 2 points cette année) mais qui reste, finalement, moins répandue que de télécharger un film : 15% des internautes l'ont fait au cours des douze derniers mois (pour mémoire, sur la même période, 24% ont téléchargé un film). Un peu plus de cinq millions de personnes regarderaient donc la télé sur Internet.

Travail et formation

Internet peut certes se révéler un formidable outil pour communiquer, se divertir, se détendre mais aussi pour travailler ! Après une légère baisse l'année dernière, la tendance repart à la hausse : 22% de l'ensemble de la population s'est cette année aidée d'un micro ou d'Internet à domicile pour Travailler Enfin, Internet permet également **de se former** à distance : 17% de l'ensemble de la population, 27% des internautes (dont 30% des actifs et 40% des élèves et des étudiants) ont utilisé ce moyen pour accroître leurs compétences, scolaires, universitaires ou professionnelles

Administration électronique

Si Internet est une porte d'entrée extraordinaire vers les loisirs ou un nouveau moyen de travailler chez soi, c'est aussi un mode de contact qui a profondément changé les relations entre les citoyens et leurs administrations. Déclarer ses impôts en ligne, vérifier les remboursements de la Sécurité Sociale, payer la cantine du petit dernier, obtenir un extrait

d'acte de naissance … Autant de démarches autrefois fort contraignantes qui se sont vues simplifiées en quelques clics Au total, 37% des Français (soit environ dix-neuf millions de personnes) jouent le jeu, ce qui correspond à 58% de l'ensemble des internautes

Commerce électronique
En ces temps d'interrogations sur le pouvoir d'achat, il est un mode de consommation qui a le vent en poupe. Acheter sur Internet est aujourd'hui, sinon un réflexe, du moins un acte banal : 38% des Français déclarent l'avoir fait au cours des douze derniers mois (+ 5 points en un an). Le nombre de personnes concernées s'élève désormais à presque vingt millions.

Le téléchargement de logiciels
Notre tour d'horizon des pratiques sur Internet se clôt avec le téléchargement de logiciels. On a déjà évoqué, plus haut, le téléchargement de musique (pratiqué par 37% des internautes) et celui de films (24%). Or le téléchargement de logiciel apparaît comme une habitude plus ancrée encore. Quatre internautes sur dix disent avoir réalisé une telle opération sur leur ordinateur au cours des douze derniers mois (Graphique 57). La proportion a progressé de 3 points en un an. Cela représente, en 2008, plus du quart de l'ensemble de la population (26%), soit un peu moins de quatorze millions de personnes.

Les freins au développement d'Internet
Les Français sont de plus en plus immergés dans l'univers d'Internet : on l'a vu, 63% se sont connectés, d'une façon ou d'une autre, sur le Web au cours des douze mois écoulés. Pour autant, cela ne les empêche pas de penser que des freins subsistent à l'utilisation d'Internet. Deux difficultés se détachent (Tableau 53), même si on va voir qu'elles ne sont pas évoquées par tous de la même façon : le **manque de protection des données individuelles** est évoqué par 20% des enquêtés (- 3 points par rapport à 2007), la **complexité** de l'outil l'est par 19% (+ 2 points en un an). Ensuite, 11% des personnes interrogées pensent qu'Internet n'est, tout simplement, « **pas utile** pour la vie quotidienne ». 10% estiment qu'Internet revient **trop cher** (- 4 points) et 10% que **le service après-vente et l'assistance laissent à désirer**.

ANNEXE II-2 : Grille d'entretien auprès des industriels

Profil :

- Hitech : Pertech, Aldebaran Robotics et Hilabs.
- Fournisseurs : FNAC (vendeurs téléphone, hifi, équipement wifi, informatique), CORRIOU JCD cuisine (cuisinistes), la compagnie de chauffage de Grenoble (chauffagistes, télé-alarmes)

Préalable aux entretiens

Avant l'entretien sur site, explorer les sites web et voir les produits commercialisés par les entreprises et s'il existe des démos de leurs produits.

Elaborer une fiche de présentation sur l'entreprise, ce qui permet de bien animer les entretiens.

Recrutement

- Le premier contact est fait par téléphone.

Bonjour, je m'appelle Emeric Fontaine, je suis étudiant en thèse à l'université de Grenoble. Dans le cadre d'une étude menée avec l'appui du CNRS sur l'intelligence ambiante, je dois rencontrer des industriels de votre domaine d'activité pour avoir des informations sur les outils qui favorisent l'habitat intelligent (avoir des exemples pour chaque industriel)
1) A qui puis-je m'adresser dans votre entreprise ? Pourriez-vous me mettre en relation ou me fournir ses coordonnées? (sans oublier le merci)
2) Afin d'aborder les thèmes suivants :

- Votre société et ses produits
- Votre clientèle vis-à-vis des aspects programmation et interconnexion de vos produits
- Votre impression sur l'évolution à plus ou moins long terme de vos produits
- Démonstration si possible de vos produits

Est-il possible de nous rencontrer?

Grille d'entretien

Notre étude a pour objectif de connaître les habitudes et les comportements des usagers pour organiser leur espace familial, de connaître les équipements de leur habitat sur lesquels ils peuvent programmer ou automatiser des fonctions, de connaître leurs souhaits vis-à-vis des objets du quotidien qui peuvent communiquer et interagir. Dans ce cadre, notre étude se décline en plusieurs étapes. L'une d'elle est de rencontrer des professionnels ayant un rapport à l'intelligence ambiante ou à l'habitat programmable. L'objectif est de faire un état des technologies actuelles en intelligence ambiante, de mesurer les besoins du grand public face à ces nouveaux outils, d'avoir un premier aperçu de ce que les personnes possèdent chez elles, de connaître les retours utilisateurs sur ces produits.

L'entretien va se dérouler en trois temps :

Pour les industriels Hi-tech

L'entretien débute par des questions relatives à votre société et vos produits sans entrer dans les détails techniques. La deuxième partie portera sur les habitudes de vos clients en termes de programmation et d'interconnexions de dispositif. La troisième partie s'intéresse à votre avis sur les évolutions des équipements du quotidien et leurs interconnexions. Enfin, si cela est possible, et sans atteindre à la confidentialité de vos produits, nous souhaiterions avoir une démonstration.

- **La société et les produits**

- Pouvez-vous me raconter rapidement comment votre société a été créée ?
- S'il y a une interaction HM originale, comment a-t-elle été imaginée … ?
- Quels sont les produits que vous commercialisez aujourd'hui?
- Quel est le niveau d'interopérabilité entre les outils que vous développez ? Niveau de connexion ?
- La connexion pour quels types de services ? (commodité immatérielle, assistance fournie à des personnes en informatique : une fonction ou ensemble de fonctions qui font sens, qui sont utiles)

- **Les clients et la programmation**

- Quels sont vos principaux clients ? utilisateurs ?
- Quels sont les retours de vos utilisateurs ? Comment avez vous amélioré le produit ?
    - Est-ce plutôt du fonctionnel (utile)
    - Est-ce de l'IHM ? (utilisable)
- Et par rapport aux fonctions de programmation, quelles sont les préférences de vos clients ?
- Pour eux, quels sont les avantages de ces outils programmables ?
- A votre avis, quels sont les principaux freins des dispositifs programmables ?
- A votre avis, pourquoi les clients n'aiment pas programmer ?
- Que programment-ils ? Pourquoi ils aiment ça ?
- Que pensent les clients des télécommandes des produits ?
- Quels sont les retours de vos utilisateurs par rapport à la programmation des dispositifs ?
    - Est-ce plutôt du fonctionnel (utile)
    - Est-ce de l'IHM ? (utilisable)

- **Les clients et l'interconnexion**

- Est-ce que vos clients souhaitent parfois interconnecter des dispositifs ? si oui lesquels ?
- Question réservée à la hi-fi, télécom : A votre avis, vos clients sont-ils plutôt des **constructeurs à façon ou préfèrent-ils les dispositifs clés en main ?**
- Est-ce que parfois vos clients vous demandent des dispositifs ou services qui sortent de l'ordinaire, non prévu au catalogue ? Pouvez me citez un exemple ? Comment répondez-vous à leur demande ?

- **Evolution des produits et de l'interconnexion**

- A votre avis, quelles sont perspectives d'évolution des produits dans un futur proche ?
- Et à plus long terme ?
- Et en termes de connexions des dispositifs, quelles évolutions ?
- Que pensez-vous d'un protocole commun en sorte que des dispositifs échangent et se comprennent ?
    - En termes techniques
    - En termes fonctionnels
    - En termes IHM

- **Démonstration**

- J'ai trouvé ces démonstrations sur votre site, en avez-vous d'autres ? Est ce que nous pourrions les utiliser pour nos expériences futures ?
- Avez-vous des démos filmées de vos produits ? Pourriez-vous me montrer des démos ? Pourrions-nous les utiliser pour nos expériences futures ?

- L'entretien est terminé, …..

  Pourrions-nous rencontrer d'autres personnes de votre société ? ou d'autres personnes dans d'autres sociétés ?

Pour les industriels « appliances »

- L'entretien débute par des questions relatives aux produits ou services vendus par votre société. La deuxième partie portera sur les habitudes de vos clients en termes de programmation et d'interconnexions de dispositif. La troisième partie s'intéresse à votre avis sur les évolutions des équipements du quotidien et leurs interconnexions. Enfin, si cela est possible, nous souhaiterions avoir une démonstration.

- **Les produits de la société**

- Quels sont les principaux produits que vous vendez aujourd'hui? Le top ten des ventes ?
- Selon vous, pourquoi sont-ils dans le top ten ? (spontané voir si la **programmation** et si l'**interconnexion** entre les outils sortent)

- **Les clients et la programmation**

- Quand vous installez ces produits, est ce que les clients attendent des fonctions de programmation ? automatisation ?
- Et par rapport aux fonctions de programmation quelles sont les préférences de vos clients ?
- Quels types d'aide attendent-ils ? Lesquels apportez-vous ?
- Que programment-ils ? Pourquoi ils aiment ça ? Pourquoi ils n'aiment pas ça ?
- A votre avis, à quelle fréquence le font-ils ?
- Pour eux, quels sont les avantages de ces outils programmables ?

- A votre avis, quels sont les principaux freins des dispositifs programmables ?
- En général, que pensent les clients des commandes à distances, (télécommande ou commande plus lointaine) des produits ?
- Quels sont les retours de vos utilisateurs par rapport à la programmation des dispositifs ?
    - Est-ce plutôt du fonctionnel (utile) ?
    - Est-ce de l'IHM ? (utilisable)

- **Les clients et l'interconnexion**

- Est-ce que vos clients souhaitent parfois interconnecter des dispositifs, que ces dispositifs soient achetés chez vous ou ailleurs ? Si oui lesquels ?
- Question réservée à la hi-fi, télécom-réseau: **A votre avis, vos clients sont-ils plutôt des constructeurs à façon ou préfèrent-ils les dispositifs clés en main ?**
- Est-ce que parfois vos clients vous demandent des dispositifs ou services qui sortent de l'ordinaire, non prévus au catalogue. Pouvez-vous citer un exemple ? Comment répondez-vous à leur demande ?

- **Evolution des produits et de l'interconnexion**

- A votre avis, quelles sont les perspectives d'évolution des produits dans un futur proche ?
- Et à plus long terme ?
- Et en termes de connexions des dispositifs, quelles évolutions ?
- Que pensez-vous d'un protocole commun en sorte que des dispositifs échangent et se comprennent ?
    - En termes techniques
    - En termes fonctionnels
    - En termes IHM

- **Démonstration**

- *Pourriez-vous me montrer des démos ?*
- *Si autres démos non connues puis-je les utiliser ?*

L'entretien est terminé, …..

Pourrions-nous rencontrer d'autres personnes de votre société ? ou d'autres personnes dans d'autres sociétés ?

ANNEXE II-3 : Scénario directeur pour le développement de KISS

La situation 1 est conçue à partir des résultats de l'étude de terrain. Elle rejoint les autres du scénario prospectif du projet ANR CONTINUUM

Situation 1 : le matin à la maison

MGenius(compagnon autonome d'aide à la vie de tous les jours) sait que c'est le jour de départ en vacances et que Bob souhaite se lever à 6h50 (T=0). Il est 6h30, 20 minutes avant l'heure du réveil (T=-1), MGenius augmente le chauffage des pièces dans lesquelles Bob va circuler. Pour des raisons d'économie d'énergie, ces pièces sont régulées à une température de 18°C la nuit et à 21°C quand Bob y circule.

Il est 6h50 (T=0), l'heure de se lever. Comme Bob est encore au lit et qu'il a demandé à être réveillé, MGenius active le système de réveil : il ne trouve pas le réveil (qui a dû être débranché) et choisit d'allumer le téléviseur situé dans la chambre en le réglant sur une chaîne musicale. La musique se met en route à faible volume, puis une lumière douce s'allume. Bob ne semblant pas se réveiller, MGenius, conformément aux préférences de Bob, emploie les grands moyens : l'intensité de chaque signal augmente. Bob s'éveille et appuie sur le bouton situé sur la table de chevet et se lève. Ce bouton tient lieu de confirmation à MGenius, il lui confirme qu'il est éveillé et qu'il souhaite que la routine du matin se déclenche. MGenius met une musique douce avec un volume sonore acceptable baisse la luminosité.

Bob enfile ses chaussons et prend le chemin des toilettes. (T=1), pendant ce temps, l'eau de la douche préchauffe. Bob se rend ensuite dans la salle de bain, la musique douce provenant de la télé migre sur la radio de la salle de bain, qui s'éclaire alors de façon tamisée (T=2). Une lumière projetée sur le plafond au-dessus de la douche lui indique la température de l'eau. A cet instant, la lumière est orangée, ce qui signifie que la température est telle que Bob l'apprécie. Il prend sa douche, MGenius augmente la lumière progressivement et passe la musique dans un registre un peu plus dynamique.

Bob sort de la douche, prend sa serviette et commence à se sécher. En parallèle, MGenius déclenche la cafetière de sorte que le petit-déjeuner soit prêt, car Bob met en moyenne 4min pour s'habiller et aller jusqu'à la cuisine. 4min est le temps nécessaire pour que l'eau de la cafetière chauffe, que le café coule complètement et, qu'il se soit refroidi juste ce qu'il faut pour qu'il soit buvable.

Bob une fois sec se rend dans sa chambre devant son armoire à vêtements (T=3). La musique re-migre vers la télé de la chambre. Cette armoire dispose d'un grand miroir qui affiche un ensemble vestimentaire en adéquation avec l'activité du jour, la météo, le contenu du placard et les préférences de Bob. Les préférences de Bob sont « montre moi la tenue que je portais lors d'une situation similaire, ou si impossible, une tenue s'en rapprochant ». MGenius affiche conformément à

la volonté de Bob une tenue vestimentaire via ce miroir. Bob n'en étant pas complètement satisfait en particulier du pull proposé, fait alors un geste de la main pour changer le pull. Une fois satisfait de son choix, il ouvre son armoire ; les vêtements qu'il a repérés à l'aide du miroir sont mis en valeur grâce à des jeux de lumière. MGenius a tiédi l'ensemble des vêtements grâce à l'armoire chauffante. Bob les enfile et se dirige à la cuisine pour y prendre son petit déjeuner.

Pendant son petit-déjeuner, MGenius informe Bob des conditions météo et de circulation (trafic, travaux) sur son trajet (T=4) via la télévision de la cuisine, puis lui met une radio d'information, car Bob aime boire son café en écoutant les infos. Le téléphone sonne alors qu'il se brosse les dents. Comme l'appel n'est pas urgent (interlocuteur non central dans la vie de Bob), MGenius décroche pour Bob et enregistre le message (T=5). Bob à programmé le comportement du répondeur à la façon central d'appel. La lumière n'est plus tamisée, la musique est dans un registre tonique. Bob face à son miroir de salle de bain en se brossant les dents voit les tâches à faire aujourd'hui. Ces tâches sont extraites de son agenda et de ses mémos. La liste est affichée sur le coin en haut à gauche du miroir. Ce couplage à été programmé par Bob ainsi que le lieu et la présentation des rappels. Il voit qu'il doit appeler Alice dès qu'il est sur la route, qu'il ne doit pas oublier de prendre le cadeau qu'il a acheté pour Alice, qu'il doit penser à fermer la fenêtre de la salle de bain. Il peut aussi utiliser le miroir pour gérer ses mails, visualiser l'état de son habitat et d'autres choses encore, mais là Bob n'a plus le temps. Il finit de se brosser les dents et demande à MGenius de lui préparer une sélection de musiques, de films et de jeux qu'il utilisera pendant ses vacances.

Bob va à sa chambre finir ses préparatifs, son sac, prend le cadeau d'Alice (T=6) et se rend dans l'entrée de son domicile pour partir.

Les clefs de la maison sont dans une corbeille dans l'entrée. Au-dessus de cette corbeille il y a le tableau d'un jeune artiste qui plait énormément à Bob. Il le regarde tous les jours avant de partir. MGenius a repéré que Bob n'a pas fermé la fenêtre de la salle de bain, il se sert du support vitré du tableau pour afficher cette information à Bob. MGenius suite aux préférences de Bob affiche tous les alertes et rappels : par exemple des robinets mal fermés, des équipements restés alimentés, ainsi que la liste des tâches à faire non encore réalisées. (T=7). D'ailleurs nous pouvons y voir clairement écrit : « tu as oublié ton portefeuille ». Bob corrige tous ces petits oublis et saisit la poignée.

Quand Bob passe la porte, tout est vérifié, MGenius verrouille la maison, redirige l'interphone sur le WeCo de Bob et active l'alarme. Le système de détection de présence bascule en mode absence (Bob part pour 1 mois). En cas d'intrusion, il avertira la société de gardiennage ainsi que Bob par tous les moyens possibles et le plus rapidement possible.

Situation 2 : entre la maison et la voiture à l'arrêt

Bob se dirige vers sa voiture. MGenius signale sur son WeCo qu'il a manqué un appel alors qu'il était dans la salle de bain. Bob demande à écouter le message. Tout en l'écoutant, il s'installe au volant. Au moment où Bob pose son WeCo, mGenius bascule la communication sur les haut-parleurs et active le micro pour que Bob puisse rappeler son interlocuteur. La conversation terminée, Bob démarre. MGenius met en route le GPS et rappelle les conditions météo sur le trajet au moyen des ressources de la voiture.

Situation 3 : en route

Lorsque les conditions de circulation sont difficiles, mGenius bascule les communications téléphoniques sur le répondeur. En chemin, Bob réalise qu'il a oublié les photos qu'il devait montrer à son amie Alice. Il demande à mGenius de rapatrier les photos « laissées » à la maison. Il n'a pas besoin de s'authentifier. Ceci est fait par l'usage-même de mGenius depuis la voiture.

Bob, sensible au respect de l'environnement, prend l'autoroute du Soleil classée « voie verte ». Il est automatiquement pris en charge par le service de régulation du trafic routier qui gère les conditions de circulation et de pollution (via des capteurs humidité, vitesse du vent, …). Des messages sont affichés sur les panneaux de signalisation (nouvelle vitesse à respecter, accidents, bouchons, pic de pollution, etc.). MGenius, conformément au choix de Bob, affiche ces mêmes informations sur le dispositif le plus approprié (tableau de bord, haut-parleurs, GPS, etc.) et surtout, rappelle les consignes de vitesse lorsque Bob ne les respecte pas.

Après Montélimar, Bob s'inquiète de la présence d'une fumée blanche dans le ciel sur sa droite. Il demande à mGenius de l'informer sur cette pollution atmosphérique. Il obtient très vite la réponse: il s'agit simplement de l'évaporation issue de la centrale de Tricastin.

Situation 4 : pause déjeuner

Vers midi, mGenius propose une liste de restaurants conformes aux préférences de Bob (gastronomie, prix, distance, confort, etc.). Bob, ayant décidé de bien profiter de ses vacances, choisit un restaurant gastronomique assez proche de l'autoroute. Après un agréable repas, il décide de régler la note avec son service de paiement bancaire automatique. Une authentification forte est requise pour utiliser le porte-monnaie électronique. Pour cela, Bob doit fournir une preuve supplémentaire de son identité. Compte tenu du lieu public, Bob ne peut pas utiliser la reconnaissance vocale. MGenius lui suggère d'utiliser le lecteur biométrique du WeCo ou de composer le mot de passe au moyen du clavier. Bob reprend la route.

Situation 5 : dans le hall de l'hôtel

Bob arrive à l'hôtel. Il pourrait se diriger directement vers sa chambre : dès le parking de l'hôtel, mGenius lui indique toutes les informations utiles pour gagner sa chambre et la clé numérique de la chambre est automatiquement transférée sur le WeCo de Bob. Toutefois, Bob préfère s'adresser à l'hôtesse car il aimerait être conseillé sur des activités de villégiature. Il y a la queue à la réception. Pour passer le temps, il consulte les informations touristiques publiées sur le grand mur du hall. Comme il est seul devant le mur, il peut naviguer dans l'espace d'informations et sauvegarde sur son WeCo quelques adresses intéressantes (curiosités, visites de musée, randonnées, loisirs sportifs, restaurants, etc.). Quelqu'un arrive et s'intéresse au mur : les informations murales sont maintenant disponibles sur le WeCo de sorte que Bob puisse continuer son exploration en privé. Arrive son tour. Son exploration des curiosités est suspendue.

Bob, proche du comptoir, est automatiquement identifié sur le PC de l'hôtesse qui lui confirme sa réservation. Comme Bob a permis à mGenius d'exporter les bonnes adresses qu'il vient de relever,

l'hôtesse propose de lui réserver une activité de son choix. La réservation est confirmée par mGenius sur le WeCo de Bob.

Situation 6 : vers la chambre d'hôtel

Satisfait, Bob se dirige vers sa chambre. MGenius perçoit cette action et règle d'ores et déjà le confort de la chambre.

Bob, à une dizaine de mètres de sa chambre, constate qu'une porte s'éclaire, lui facilitant ainsi le repérage de sa chambre. Cependant une autre personne le suit à quelques pas, et c'est en entendant son WeCo biper devant la porte qu'il a la confirmation qu'il s'agit bien de sa chambre. La porte s'ouvre automatiquement grâce à la clef électronique récupérée à l'arrivée. Bob entre.

Situation 7 : dans la chambre d'hôtel

MGenius a déjà réglé l'ambiance lumineuse et sonore de la chambre en respectant autant que possible les préférences de Bob. Il a notamment affiché la photo préférée qui rappelle le bon pays de Grenoble ! Les services disponibles sont automatiquement indiqués y compris l'exploration des informations touristiques commencée dans le hall de l'hôtel. Il réserve une table pour deux personnes au restaurant de l'hôtel.

Exténué par le voyage, il s'allonge et s'endort. Alice va bientôt arriver. MGenius le sait et réveille Bob avec les ressources disponibles… Il est l'heure d'accueillir Alice dans le hall de l'hôtel. Plus tard, ils iront tous les deux au restaurant.

Situation 8 : au restaurant

Il y a du monde. Entre deux plats, Bob propose de montrer à Alice ses dernières photos de montagne. Pour cela, le weco n'est pas très commode. Mgenius rend les photos disponibles sur la table. Tous deux explorent les photos sur la table. Le service utilise n'est pas tout à fait le même que celui de la maison.

Alice trouve la photo du glacier de celliers particulière car elle lui rappelle celle de la Muzelle qu'elle a photographiée l'année dernière. Elle la récupère sur son weco et met à disposition celle de la Muzelle sur la table. Parce que bob et Alice sont amis, ces échanges sont permis en toute transparence et sécurité…

En fin de repas, bob demande à Mgenius de lui fournir la note de restaurant qu'il valide. Comme il ne veut pas montrer l'addition a son amie, l'affichage a lieu sur le weco, non pas sur la table. Apres ces délicieux instants passes avec son amie, bob rejoint sa chambre …

Situation 9 : incident médical

Dans la nuit, Bob montre des signes alarmants de santé. Personne à qui se renseigner. Il fait appel à mGenius. Celui-ci lui signale que Grenoble est à 5h de route en voiture. Soit il rentre sur Grenoble, soit il consulte sur Vallauris. Les vacances étant planifiées depuis longtemps, il préfère rester sur

Vallauris. MGenius lui propose plusieurs solutions : le médecin de garde, les urgences à l'hôpital le plus proche, etc. Il opte pour le médecin de garde. MGenius le met en contact avec le médecin via son SmartPhone. Puis, Bob ayant obtenu un rendez-vous en urgence, un plan est affiché pour s'y rendre sur son WeCo. La pharmacie de garde est également indiquée et localisée. Alice est avertie pour qu'elle le conduise au bon endroit.

ANNEXE IV-1 : Grammaire de KISS

Grammaire correspondant au scenario Bob Domus.

Meta-Grammaire :
ABNF
+ { *sémantique* } ; utilisé pour la création d'AA
+ ! *cross Référence* ! ; utilisé pour pointer sur une autre grammaire ou sur la BDC
+ ? *dépendance* ; utilisé pour demander des informations supplémentaire dans la sémantique

Sémantique :
CONDITION: localisation dans la grammaire de la structure condition
ACTION: localisation dans la grammaire de la structure action
<1> : index du 1er NT dans la grammaire
LINK : si LINK n'est pas précédé d'un index, l'action est instantanée, sinon elle est durable.
PC : ajoute une MetaData spécifique dans le point de coupe généré
PARAM_STRING : permet d'envoyer des paramètres de type string, les composants d'un param doivent être décoré par des " .
PARAM_STRING "a" : la sémantique du niveau supérieur donne la méthode qui convient avec ce paramètre
"a" PARAM_STRING "b" : a est la méthode qui prend en paramètre b
PARAM_INT : idem, mais avec des paramètres int.
PARAM_BOOL: idem, mais avec des paramètres boolean.
/ : correspond à l'alternation de la grammaire.
INST : indique que la condition est instantanée
LASTING : indique que la condition est durable
EQUAL, SUP, INF : des operateurs de condition
TOBOOL : operateur de transformation event to boolean.
X-X : séparateur de condition sémantique

Grammaire noyau : coreGramBobDomusV1.grm :

```
sentence=condition action {CONDITION:<1> ACTION:<2>}
condition=fact [operator condition]
action=unitaryAction [operator action]
operator="et "

unitaryAction=actionName

fact=!contextualInfoGram!
actionName=!actionGram!
```

Grammaire d'action : actionGramBobDomusV1.grm

```
actionName=wakeUp / open / close / listen / switchOn / switchOff / watch / heatUp
wakeUp="être réveillé avec " alarm  {LINK<1>}
open="ouvrir " opennable {LINK<1>}
close="fermer " closable {LINK<1>}
```

```
listen="ecouter " listenable {LINK<1>}
switchOn="allumer " switchOnAble {LINK<1>}
switchOff="eteindre " switchOffAble {LINK<1>}
watch="watch" observable "sur " viewer {<1>LINK<2>}
heatUp="chauffer " heatAble {LINK<1>}

alarm=!abstractThingsGram!
opennable=!abstractThingsGram!
closable=!abstractThingsGram!
listenable=!abstractThingsGram!
switchOnAble=!abstractThingsGram!
switchOffAble=!abstractThingsGram!
observable=!abstractThingsGram!
viewer=!abstractThingsGram!
heatAble=!abstractThingsGram!
```

Grammaire de classe abstraite : abstractThingsGramBobDomusV1.grm

```
alarm=lightAlarm/audioAlarm
lightAlarm=switchOnAbleLight
audioAlarm="une musique " soundLevelParam switchOnAbleAudio {"SetVolume"
PARAM_INT "0"}

switchOnAbleLight=("les lumieres " (lightLocation / lightIntensityLow /
lightIntensityMedium / lightIntensityStrong ) / (lightLocation lightIntensityLow /
lightIntensityNull / lightIntensityMedium / lightIntensityStrong ) ) {"SetTarget"
PARAM_BOOL "True"}
lightIntensityLow="avec une intensite faible " {"SetLoadLevelTarget" PARAM_INT "20"}
lightIntensityMedium="avec une intensite moyenne " { "SetLoadLevelTarget" PARAM_INT
"50"}
lightIntensityStrong="avec une intensite forte " {"SetLoadLevelTarget" PARAM_INT "90"}
lightIntensityNull="avec une intensite null " {"SetLoadLevelTarget" PARAM_INT "0"}

soundLevelParam="a volume " soundLevelLow / soundLevelNormal / soundLevelStrong
{"SetVolume" PARAM_STRING "Master"}
soundLevelLow="faible " {"SetVolume" PARAM_INT "50"}
soundLevelNormal="normal " {"SetVolume" PARAM_INT "65"}
soundLevelStrong="fort" {"SetVolume" PARAM_INT "85"}
switchOnAbleAudio=paramAudioMuteFalse0 paramAudioMute0 paramAudioMuteFalse1
{"SetMute" PARAM_INT "0"}
paramAudioMuteFalse0=" " {"SetMute" PARAM_INT "0"}
paramAudioMute0=" " {"SetMute" PARAM_STRING "Master"}
paramAudioMuteFalse1=" " {"SetMute" PARAM_BOOL "False"}

opennable="les volets " {"SetPosition" PARAM_STRING "Opened"}
closable="les volets " {"SetPosition" PARAM_STRING "Closed"}
listenable=audioAlarm
switchOnAble=switchOnAbleLight / switchOnAbleCoffe
switchOnAbleCoffe="la cafetiere " {"SetTarget" PARAM_BOOL "True"}
```

```
switchOffAble=switchOffAbleLight / switchOffAbleCoffe / switchOffAbleAudio {}
switchOffAbleLight=("la lumiere " [lightLocation ] ) {"SetTarget" PARAM_BOOL "False"}
switchOffAbleCoffe="la cafetiere " coffeMachine {"SetTarget" PARAM_BOOL "False"}
switchOffAbleAudio="la musique " paramAudioMute0 paramAudioMute1 {"SetMute"
PARAM_INT "0"}
paramAudioMute1=" " {"SetMute" PARAM_BOOL "True"}

observable=temperatureObservable
viewer=temperatureDisplay
temperatureObservable="temperature de l eau " {CurState}
temperatureDisplay="la douche " {SetValue}
heatAble="l eau de la douche " {"SetTargetTemp" PARAM_INT "25"}

LightLocation=!contextualInfoGram!
coffeMachine=!contextualInfoGram!
```

Grammaire du contexte : contextualInfoGramBobDomusV1.grm

```
fact=locationFact / timeFact / dryingFact / buttonPress / buttonOn
locationFact=locationFactLasting / locationFactInst
timeFact="il est " time
dryingFact=drying {INST<1>}
buttonPress="j'appuie sur "button {INST<1>}
buttonOn=button "a ete enclenche " {LASTING<1>}

locationFactLasting="je suis " presenceDetector {LASTING<1>}
locationFactInst=locationFactInstIn / locationFactInstOut
locationFactInstOut="je sort " presenceDetectorInstOut {INST<1>}
locationFactInstIn="je rentre " presenceDetectorInstIn {INST<1>}

presenceDetector=presenceDetectorIn / presenceDetectorOut
presenceDetectorIn=sensorLitIn / sensorBathroomIn / sensorBedroomIn / sensorKitchenIn
{OccupancyStateTOBOOL"Occupied"}
presenceDetectorOut=sensorLitOut / sensorBathroomOut / sensorBedroomOut /
sensorKitchenOut {OccupancyStateTOBOOL"Unoccupied"}

sensorLitIn="dans le lit" {PC location:bed}
sensorLitOut="hors du lit" {PC location:bed}
sensorBathroomIn="dans la salle de bain" {PC location:bathroom}
sensorBathroomOut="hors de la salle de bain" {PC location:bathroom}
sensorKitchenIn="dans la cuisine" {PC location:kitchen}
sensorKitchenOut="hors de la cuisine" {PC location:kitchen}
sensorBedroomIn="dans la chambre" {PC location:bedroom}
sensorBedroomOut="hors de la chambre" {PC location:bedroom}

presenceDetectorInstIn=sensorLitIn / sensorBathroomIn / sensorBedroomIn / sensorKitchenIn
{OccupancyStateTOBOOL"Occupied"}
presenceDetectorInstOut=sensorLitInstOut / sensorBathroomInstOut / sensorBedroomInstOut
/ sensorKitchenInstOut {OccupancyStateTOBOOL"Unoccupied"}
```

```
sensorLitInstOut="du lit" {PC location:bed}
sensorBathroomInstOut="de la salle de bain" {PC location:bathroom}
sensorKitchenInstOut="de la cuisine" {PC location:kitchen}
sensorBedroomInstOut="de la chambre" {PC location:bedroom}

time=clockTime/betweenTime
clockTime=clock {INSTTimeEQUAL<1>}
betweenTime="entre " clock "et " clock {INSTTimeSUP<1>X-XTimeINF<2>}
clock="21h" / "22h" / "07h05" / "09h" / "07h30" / "07h10" / "20h" / "10h" / "08h" / "07h"
{2100/2200/705/900/730/710/2000/1000/800/700}
button="le bouton " {ValueTOBOOL"0"}

drying="je me seche" dryingTowel {OccupancyStateTOBOOL"Occupied"}
dryingTowel=" " {PC location:towel}

lightLocation=bathroomLocation / bedroomLocation / kitchenLocation
coffeeMachine=" "
```

ANNEXE V-1 : Déroulement de l'expérimentation

- **Accueil et signature du consentement**

Bonjour,
Tout d'abord merci d'avoir bien voulu participer à notre expérience. Nous allons vous faire tester une application qui permet de piloter un habitat intelligent. C'est-à-dire un habitat dans lequel vous allez pouvoir programmer différents objets. Notre application est encore en phase d'évaluation et c'est pour cela que nous avons besoin de vos retours aussi bien positifs que négatifs. Tous vos commentaires sont importants pour que nous puissions améliorer l'application.
Afin de pouvoir exploiter les résultats, nous allons enregistrer ce que vous ferez sur l'ordinateur, nous allons vous enregistrer et vous filmer. Ces données sont uniquement à destination de la recherche et seront traitées de manière anonyme puis détruites une fois réalisée l'analyse de ces données (dans les 3 mois). Pour être conforme à la législation, nous vous demandons, si vous l'acceptez, de signer ce consentement pour enregistrer ces données expérimentales.
EXPERIMENTATEUR : FAIRE SIGNER LE CONSENTEMENT

- **Faire visiter l'appartement : DONNEES RECUEILLIES : VIDEO ET AUDIO**

Nous allons maintenant nous rendre dans l'appartement « DOMUS ». Je vais vous faire visiter cet appartement et vous montrer les différents capteurs qu'il possède et les services qu'il offre. Tout d'abord, les pièces sont munies de capteurs de présence, c'est-à-dire que le système sait dans quelle pièce vous vous trouvez.

EXPERIMENTATEUR : donner au participant un plan de l'appartement avec tous les capteurs et effecteurs.

Dans la cuisine, vous avez des capteurs de présence, les volets sont programmables. Les lumières et la musique sont également programmables dans cette pièce. La cafetière peut aussi se déclencher de manière automatique selon les moments de la journée.
Dans la chambre, le lit est équipé d'un capteur qui détecte votre présence dès que vous vous êtes assis dessus. Il y a aussi un capteur qui vous détecte dans la pièce. Les volets de la chambre sont programmables. Comme dans la cuisine, les lumières et la musique sont programmables.
Sur la table de nuit, vous remarquez ce bouton qui permet d'enclencher certaines fonctions. En particulier, le matin il peut jouer le rôle de « réveil matin de la maison » et lancer un ensemble de tâches routinières qui se passent le matin : ouvrir les volets, chauffer l'eau de la douche à la bonne température, allumer la cafetière, …
Dans la salle de bain, Il y a également un capteur de présence général et un capteur qui détecte votre présence dans la douche. La température de l'eau de la douche s'affiche sur le mur. Les lumières sont programmables. Et le pose-serviette est assez surprenant car il détecte la présence ou non d'une serviette.
Dans le bureau, il y a également certaines fonctionnalités mais nous les avons désactivées de manière à ne pas perturber l'expérience.

Avez-vous des questions à nous poser concernant les capteurs ou les capacités de cet appartement ?

- **Présenter et faire tester l'application en langage naturel : DONNEES RECUEILLIES: VIDEO ET AUDIO**

EXPERIMENTATEUR explique le fonctionnement de l'application, il fournit à l'utilisateur un guide d'utilisation simple de l'application.
EXPERIMENTATEUR fait un premier test avec la phrase « Allumer une lampe quand je suis dans la salle de bain », montrer le résultat en jouant le scénario dans Domus.
Maintenant, vous allez faire un exercice pour vous familiariser avec Domus en écrivant la phrase « Allumer une lampe quand je suis dans la salle de bain entre 20h et 22h ».
A vous ….
Nous allons aller voir ce qui se passe dans la salle de bain.
EXPERIMENTATEUR : constater que rien ne se produit car nous ne sommes pas dans le bon créneau horaire et montrer le fonctionnement de l'horloge virtuelle.
Faire modifier le créneau horaire par le participant.
Ok, donc nous allons voir si Domus a compris (retourner dans la salle de bains et refaire le test)

- **Présenter et faire tester Domus virtuel en 3D : DONNEES RECUEILLIES: VIDEO ET AUDIO**

EXPERIMENTATEUR : Voici une réplique virtuelle de Domus…. présenter chacune des pièces, montrer les différents capteurs et les différents outils présentés lors de la visite. (Expliquer la différence entre la vision bureau et le salon)
Nous allons tester Domus Virtuel avec la phrase « Allumer une lampe quand je suis dans la salle de bain entre 20h et 22h » . On voit que cela marche aussi dans le Domus Virtuel et dans le Domus réel.

- **Debriefing pour lever les principaux problèmes IHM rencontrés avec l'application : DONNEES RECUEILLIES: VIDEO ET AUDIO**

Notre application est en phase d'évaluation donc n'hésitez pas à nous faire part des points qui n'ont pas été clairs pour vous. Peut-être que je suis allé un peu vite et que je n'ai pas été très clair dans mes explications.
Avez-vous des questions à me poser concernant l'appartement ?
Voulez-vous le revisiter ?
Avez-vous des questions à me poser concernant le langage naturel ?
Voulez vous refaire un test ?
Avez-vous des questions à me poser concernant le Domus virtuel ?
Voulez vous refaire un test ?
Avez-vous des questions à me poser concernant l'horloge virtuelle ?
Voulez vous refaire un test ?

- **Faire utiliser le langage naturel, 3D et Domus par le sujet : DONNEES RECUEILLIES : TRACES, VIDEO ET AUDIO - ANNOTATION DES COMPORTEMENTS**

Nous allons maintenant vous demander de programmer l'habitat. Pour cela nous allons vous lire un scénario (donner le scénario papier au participant) que vous devrez programmer avec le langage naturel et ensuite vérifier si Domus Virtuel vous a compris. Je resterai auprès de vous tout au long de l'expérience.

SCENARIO A LIRE

Vous habitez dans cette maison intelligente qui contient des objets programmables : machine à café, volets, lumière, capteurs de présence, pose-serviettes, etc. Grâce à ces nouveaux dispositifs, vous allez pouvoir programmer votre maison et en particulier l'enchaînement des tâches du matin. Sur la table de chevet, un bouton vous permet de lancer la routine du matin.

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.

Tout n'est pas forcément envisageable ?

Nous vous demandons de faire cette programmation et d'expliquer à haute voix ce que vous faites et pourquoi. Vous pouvez utiliser le langage naturel, le simulateur et l'horloge virtuelle.

Quand vous pensez avoir fini, dites-le nous. Si vous n'y arrivez pas, ne vous inquiétez pas le problème vient certainement de DOMUS.

EXPERIMENTATEUR : LAISSER LE PARTICIPANT FAIRE SA PROGRAMMATION, PENDANT CE TEMPS RESTER AVEC LUI ET NOTER SES ACTIONS.

- **TESTS ET MODIFICATIONS : DONNEES RECUEILLIES : VIDEO ET AUDIO ET ANNOTATIONS DES MODIFICATIONS SOUHAITEES, TRACES DES MODIFICATIONS**

Nous allons maintenant faire le bilan avec vous sur ce que vous avez réussi à faire et sur ce qui été plus difficile.

Globalement que pensez-vous de cette expérience que vous venez de faire ?
A votre avis est ce que Domus a compris ce que vous vouliez faire ? Pourquoi ?
Qu'est ce qui a été facile ?
Qu'est ce qui a été difficile ?

Maintenant, nous allons regarder ce qui se passe dans l'appartement et vous me direz ce qui correspond à ce que vous souhaitiez et si vous souhaitez faire des modifications

EXPERIMENTATEUR : FAIRE LE TEST AVEC LE PARTICIPANT NOTER CE QU'IL VOUDRAIT MODIFIER

J'ai noté ce que vous souhaitez modifier. Je vous propose de faire ces modifications sur l'application en langage naturel et de refaire des tests.

- **DEBRIEFING : DONNEES RECUEILLIES : VIDEO ET AUDIO ET PRISE DE NOTE .**

Nous allons maintenant vous posez quelques questions sur le langage naturel et l'application Domus Virtuel.
Quels sont les points forts du langage naturel ?
les points faibles ?
Qu'avez-vous pensez du vocabulaire utilisé ?
Qu'avez-vous pensez de la manière de rédiger les phrases ?
Qu'avez-vous pensez de lui indiquer des plages horaires ?
Quels sont les points forts de Domus virtuel ?
les points faibles ?
Que pensez-vous de l'utilisation de l'horloge virtuelle ?

- **Questionnaire final et SUS (usability scale Brooke) : DONNEES RECUEILLIES : AUDIO ET Questionnaire papier**

Pour terminer nous allons vous demander de remplir ce questionnaire.
Avez-vous d'autres remarques ou questions à nous poser ?

ANNEXE V-2 : Questionnaire d'évaluation

Questionnaire d'évaluation

Vous répondrez à chacune des propositions en indiquant si vous êtes tout à fait d'accord, plutôt d'accord, plutôt pas d'accord, pas du tout d'accord, ou si vous n'avez pas d'opinion.					
Il est facile de paramétrer l'intensité lumineuse des lumières	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Le vocabulaire utilisé permet de bien identifier les objets	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Les verbes d'actions sont difficiles à comprendre	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Ajouter des contraintes horaires a été facile	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Domus virtuel n'est pas réaliste	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Domus virtuel permet de retrouver facilement les objets	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai tout à fait confiance dans Domus virtuel pour tester mon programme	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Le choix de plages horaires est souple	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Avec Domus virtuel, je repère difficilement les capteurs	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai tout à fait confiance dans Domus réel et ses capteurs	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Le langage naturel est suffisant pour programmer Domus	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Les modifications en langage naturel ont été difficiles à faire	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
L'utilisation du langage naturel et de Domus virtuel est un bon moyen pour piloter le réel	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion

Les prochaines questions concernent l'application en LANGAGE NATUREL. Vous répondrez à chacune des propositions selon l'échelle d'accord.					
J'aimerais utiliser ce système fréquemment	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai trouvé le système inutilement complexe	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Je pensais que le système était facile à utiliser	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Je pense que j'aurais besoin d'une personne technique pour être capable d'utiliser ce système	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai trouvé les différentes fonctions de ce système ont été bien intégrées	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Je pense qu'il y a trop d'incohérence dans ce système	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'imagine que la plupart des gens pourrait apprendre à utiliser ce système rapidement	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai trouvé le système très difficile à utiliser	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion

Questionnaire d'évaluation

Je me sentais très confiant en utilisant le système	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'avais besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce système	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion

Les prochaines questions concernent DOMUS VIRTUEL. Vous répondrez à chacune des propositions selon l'échelle d'accord.

J'aimerais utiliser Domus Virtuel fréquemment	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai trouvé Domus virtuel inutilement complexe	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Je pensais que Domus virtuel était facile à utiliser	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Je pense que j'aurais besoin d'une personne technique pour être capable d'utiliser Domus virtuel	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai trouvé les différentes fonctions de Domus virtuel ont été bien intégrées	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Je pense qu'il y a trop d'incohérence dans Domus virtuel	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'imagine que la plupart des gens pourrait apprendre à utiliser Domus virtuel rapidement	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'ai trouvé le Domus virtuel très difficile à utiliser	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
Je me sentais très confiant en utilisant Domus virtuel	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion
J'avais besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce Domus virtuel	○ Tout à fait d'accord	○ Plutôt d'accord	○ Plutôt pas d'accord	○ Pas du tout d'accord	○ Sans opinion

Enfin, quelques questions vous concernant

Pouvez vous indiquer votre prénom [ ]

Votre année de naissance [ ]

Votre profession [ ]

Par rapport aux nouvelles technologies , vous sentez plutôt fan ou plutôt opposant ? merci de vous situer sur l'échelle ci-dessous

Opposant — Fan

Par rapport à l'application sur l'échelle fan et opposant où vous situez vous ?

Opposant — Fan

ANNEXE V-3 : KISS pour les participants et le magicien d'Oz

L'utilisateur participant au test d'acceptabilité dispose de trois outils (figure A-V-1): l'éditeur de phrase, le monde simulé et l'horloge UPnP. Le magicien, quant à lui, dispose de la BdC simulée, du simulateur de présence, du GC, et d'interfaces complémentaires (figure A-V-2).

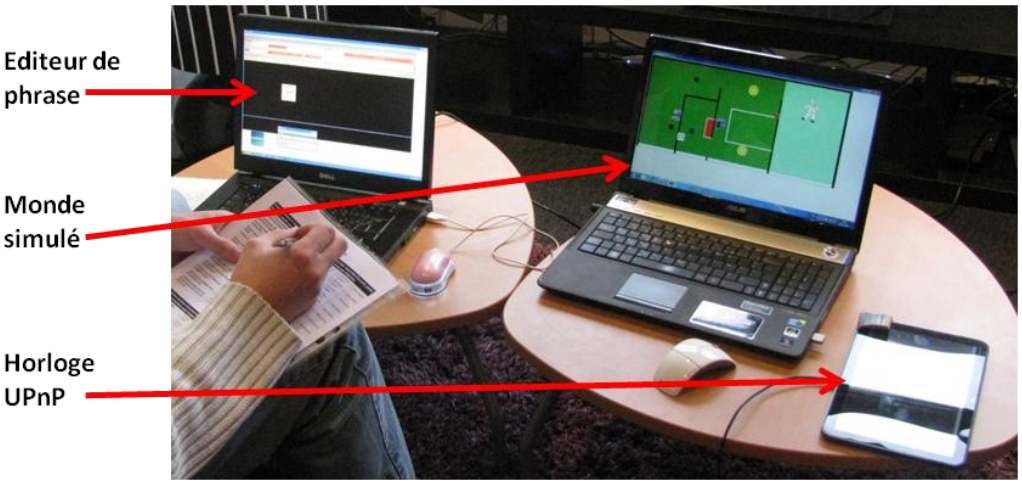


Figure A-V-1. Environnement KISS, avec de gauche à droite : l'éditeur de phrase en langage pseudo naturel, le monde simulé, et l'horloge UPnP.

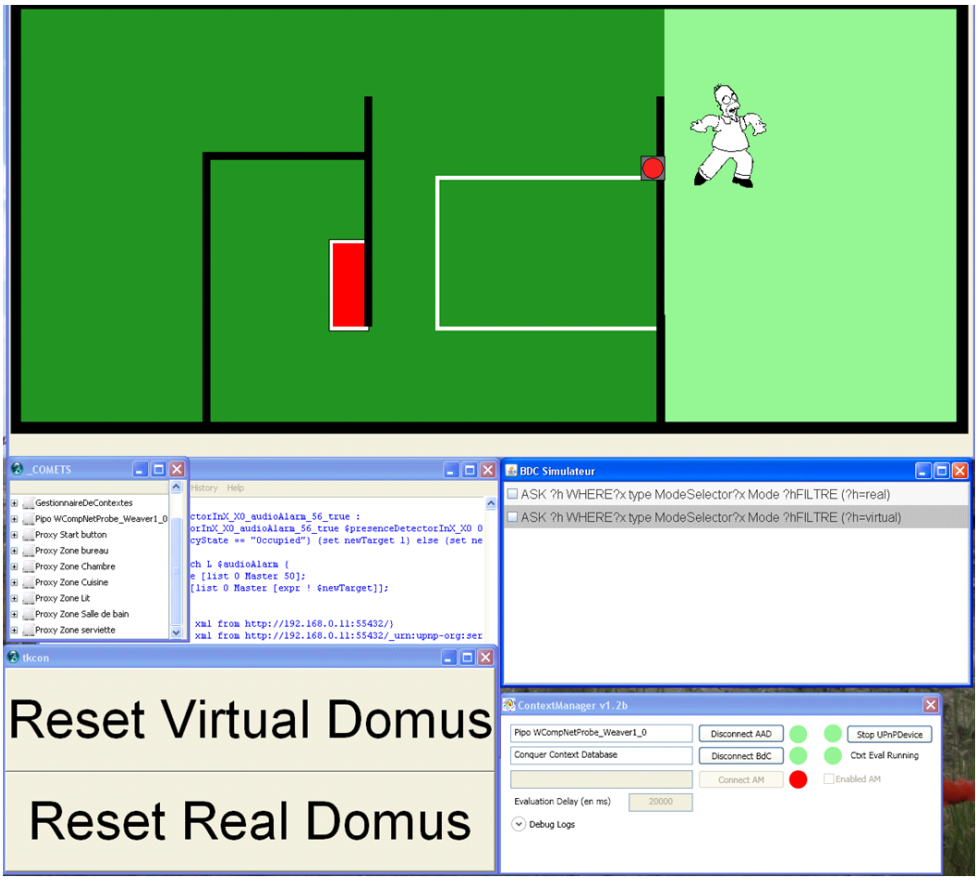


Figure A-V-2. Ecran du magicien. En haut, le simulateur de présence ; au centre droit, la BdC simulée ; dans le coin inférieur droit, le GC ; dans le coin inférieur gauche, des boutons pour réinitialiser le monde simulé ainsi que DOMUS ; à gauche au milieu, un explorateur de device UPnP et le terminal Tcl.

Résumé

Cette thèse traite du problème du développement d'espaces intelligents par l'utilisateur final sous l'angle de l'Interaction Homme-Machine et de l'Intelligence Ambiante. Dans les processus actuels de développement, l'utilisateur est un consommateur contraint par un système pensé et réalisé par d'autres. L'objectif de cette thèse est de redonner le pouvoir à l'utilisateur final par le biais d'outils adaptés au développement d'espaces intelligents. Cette thèse retient l'habitat intelligent comme lieu de vie privilégié. Ses contributions incluent : (1) *DisQo (Dispositifs du QuOtidien), une nouvelle méthode d'investigation* des besoins, réalisable au domicile de familles, qui sollicite l'imagination et assure un juste équilibre entre contrôle expérimental, respect de la sphère privée et validité écologique des résultats ; (2) Un *espace de classification* pour une lecture comparative systématique et synthétique des outils portant sur le développement et la programmation d'habitats intelligents. Cette taxonomie met en évidence le peu d'avancées en édition multisyntaxe de même pour l'aide à la mise au point de programmes ; (3) *KISS (Knit Your Ideas into Smart Spaces), un outil de programmation et de mise au point* dont le langage de programmation est de type déclaratif orienté règles, avec potentiel d'égale opportunité syntaxique entre langue française pseudonaturelle (LPN) et langage visuel iconique. La technique d'interaction de construction des programmes LPN s'appuie sur l'utilisation de menus dont les options sont calculées dynamiquement assurant ainsi la découverte progressive du langage ainsi que l'extensibilité et la correction syntaxique et sémantique des programmes. La mise au point peut se pratiquer, au choix, dans le monde physique ou dans un monde dual numérique. L'évaluation de KISS dans DOMUS, un habitat intelligent d'expérimentation, montre que les utilisateurs parviennent à programmer un scénario réaliste de la vie réelle.

Mots-clefs
Interaction Homme-Machine, intelligence ambiante, génie logiciel par l'utilisateur final, développement par l'utilisateur final, programmation par l'utilisateur final, DisQo, KISS, DOMUS.

Abstract
This dissertation addresses the problem of end-user development for smart spaces from a human-computer interaction perspective in the context of ambient intelligence. End-users are currently doomed to be consumers of systems that have been designed and implemented by others. The goal of this thesis is to provide end-users with tools that will enable them to develop their own smart spaces. This work focuses on the home as a key place for smartness. The contribution of this doctoral research is threefold: (1) *DisQo*, *a new method* for field studies that combines several techniques to reach a satisfying balance between experimental control, privacy issues, and ecological validity. Its key element is for observers to be able to "visit" people homes through the pictures of intimate objects taken by the participants themselves and to use these pictures as playful cultural probes to envision future use; (2) *A problem space* that makes explicit the functional coverage (as well as the limits) of the tools from the state of the art in the area of end-user development for smart homes. In particular, the problem space reveals a lack of support for multi-syntax editing as well as for testing and debugging programs; (3) *KISS (Knit your Ideas into Smart Spaces), an end-user development tool* that uses a declarative rule-based programming paradigm where programs are expressed in a French pseudo-natural language with potentiality for syntactic equal opportunity with an iconic visual language. Programs are constructed by selecting items from pull down menus that are dynamically updated with the functionalities of the smart home. By so doing, the end-user can learn the programming language incrementally and specify programs that are semantically and syntactically correct. Programs can be tested either in a virtual home or in a real home. The evaluation of KISS in the DOMUS experimental platform, shows that users are able to program a real-life scenario.

Keywords
Human-Computer Interaction, Ambient Intelligence, Ubiquitous Computing, End-User Software Engineering, End-User Development, End-User Programming, DisQo, KISS, DOMUS.