AppsGate, un écosystème domestique programmable : "vivre avec" comme retour d'expérience

Joëlle L Coutaz, James L. Crowley

JIPS

Journal d'Interaction Personne-Système
Volume 7, Numéro 1, Article 1, septembre 2018, pages 1 à 35
http://jips.episciences.org

Le journal de l'Association Francophone d'Interaction Homme-Machine

AppsGate, un écosystème domestique programmable : « vivre avec » comme retour d'expérience

Joëlle COUTAZ

Université Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000, Grenoble, France

joelle.coutaz@univ-grenoble-alpes.fr, james.crowley@inria.fr

James L. CROWLEY

Université Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, F-38000, Grenoble, France

AppsGate, a Programmable Domestic Eco-system: Learning from « Living in it »

Abstract. We present an experience with the development and evaluation of AppsGate, an eco-system for the home that can be programmed by end-users. We show the benefits from using the homes of the project team members as real-life living-labs. In particular, we discuss the first person perspective experience as an effective way to conduct longitudinal experiments in real world settings. We conclude that a programmable habitat is desirable provided that attention cost is minimized

Key words: End-user programming (EUP), end-user development (EUD), smart home, experimental method.

Résumé. Cet article présente un retour d'expérience avec la mise en œuvre et l'évaluation d'AppsGate, un écosystème domestique programmable par l'habitant. Nous montrons l'apport de l'utilisation des domiciles de membres du projet tout au long du processus de développement, et notamment l'intérêt de « vivre avec » comme technique d'expérimentation longitudinale.

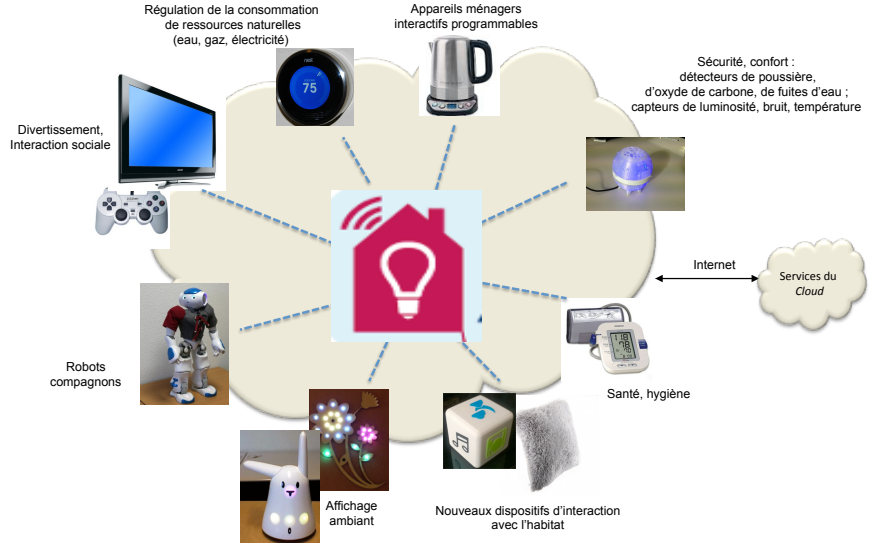
Mots-clés : programmation par l'utilisateur final (PUF), développement par l'utilisateur final (DUF), habitat intelligent, méthode expérimentale.

Édité par Joëlle Coutaz, Université de Grenoble et Patrick Girard, Université de Poitiers
Soumis le 04 août 2018 – Accepté le 30 septembre 2018

1 INTRODUCTION

L'habitat intelligent (*smart home*) fait l'objet d'une attention soutenue en recherche comme en développement industriel. Ce terme couvre de nombreuses acceptions, depuis l'argument publicitaire et une domotique évoluée fondée sur l'interconnexion d'objets et de services jusqu'à l'écosystème situé (Suchman, 1987) capable, en toute circonstance, de répondre de manière appropriée aux besoins, intentions et valeurs de l'habitant (voir Figure 1). Or, ces besoins, intentions et valeurs sont volatiles, variant d'une culture à l'autre, entre familles et individus, en fonction des circonstances et à différentes échelles de temps (Friedman et al., 1997 ; Schwartz, 1992). L'adaptation à cette variabilité individuelle et collective constitue un défi scientifique majeur pour le développement d'habitats intelligents et, par extension, de l'intelligence ambiante.

Figure 1. L'habitat intelligent comme écosystème d'objets et de services situés.



Aujourd'hui, le développement de l'habitat intelligent opère selon deux visions. Pour l'une, les habitants sont des consommateurs passifs cédant leurs données les plus privées en échange de services orchestrés par une intelligence en nuage (*cloud computing*) et contrôlés par les fournisseurs de services commerciaux. Cette approche est attrayante pour deux raisons : l'habitant est libéré de l'obligation de contrôler et de configurer les services, et les fournisseurs peuvent exploiter la puissance des données massives (*Big Data*) pour créer des services à valeur ajoutée. Le défi est d'offrir des services suffisamment attractifs pour que les utilisateurs soient prêts à abandonner leurs données personnelles.

L'alternative est que l'habitant conserve le contrôle de ses données et des services au prix d'investir l'effort nécessaire pour configurer et gérer l'écosystème. Cette approche est possible si l'écosystème est facile à utiliser et à maîtriser. L'approvisionnement collaboratif (*crowdsourcing*) est une voix prometteuse pour cette alternative. Ainsi, des communautés d'utilisateurs partageraient et perfectionneraient la conception et la programmation de l'écologie des objets et services de l'habitat. Le défi est de développer une technologie d'interaction facilement maîtrisable par les utilisateurs.

Nous avons choisi d'explorer la « Programmation par l'Utilisateur Final » (*End-User Programming*) ou PUF, voire le « *Développement par l'Utilisateur final* » (*End-User Development*) ou DUF (Lieberman et al., 2006 ; Nardi, 1993 ; Hinckley 2017) comme début de réponse à ce défi. Cette approche appelle les questions de recherche suivantes :

QUESTIONS DE RECHERCHE :

- ✓ **La Programmation par l'utilisateur final (PUF) répond-elle au problème de l'adaptation à la variabilité et à la diversité des valeurs humaines dans le cadre de l'habitat résidentiel ?**
- ✓ **Si tel est le cas, un écosystème programmable est-il désirable sachant que la motivation première de l'habitant n'est pas nécessairement d'apprendre à programmer, mais de résoudre un problème personnel qui serait plus coûteux à résoudre sans l'aide du système ?**
- ✓ **Les méthodes d'évaluation usuelles de l'Interaction Homme-Machine (IHM) suffisent-elles pour juger de l'adéquation de l'approche PUF en matière d'écosystème domestique ?**

Dans cet article, nous présentons notre expérience portant sur la réalisation et l'évaluation d'AppsGate, un écosystème domestique programmable par l'habitant. Les leçons que nous en tirons concernent d'une part, l'intérêt de l'approche PUF, d'autre part, le processus expérimental de développement et d'évaluation. Elles se résument comme suit :

APPORTS DE CETTE RECHERCHE

- ✓ **Façonner son habitat par programmation procure un sentiment réconfortant de réussite en coévolution naturelle avec l'habitat mais demande d'y consacrer du temps.**
- ✓ **Une juste coopération entre programmation par l'habitant et apprentissage automatique devrait réduire l'investissement attentionnel tout en gardant le contrôle et le pouvoir créatif souhaités.**
- ✓ **L'appréciation de la valeur ajoutée ou des insuffisances profondes d'un écosystème domestique, voire de tout système relevant de l'intelligence ambiante, exige un déploiement en conditions réelles et sur la durée.**
- ✓ **Toute expérimentation longitudinale *in situ* d'un système à usage domestique implique d'une part, le recrutement d'une bonne dizaine de familles qui acceptent de jouer le jeu durablement, et d'autre part la réalisation d'un système fiable. Il en résulte un coût élevé, souvent incompatible avec les conditions de financement et de développement de la recherche.**
- ✓ **Nous proposons le compromis suivant : effectuer chez des utilisateurs représentatifs des expérimentations courtes (un mois) complétées par des déploiements longs (une année pour couvrir les 4 saisons) dans les habitats de membres de l'équipe de développement. Après tout, *si ça ne marche pas chez les concepteurs, comment pourrait-il en être autrement ailleurs* ?**

Cet article est une présentation étendue de la publication *Evaluation d'écosystème domestique programmable : oser « vivre avec » comme méthode expérimentale »* parue dans les actes de la 29ème conférence francophone sur l'Interaction Homme-Machine, IHM'17, à Poitiers (Coutaz & Crowley, 2017). Nous proposons ici une analyse des méthodes de conception pour l'habitat résidentiel ainsi qu'une synthèse des systèmes PUF appliqués à l'habitat (Sections 2 et 3 respectivement). Ces analyse et synthèse, qui permettent de situer AppsGate dans l'état de l'art, ouvrent la discussion sur les éléments fonctionnels essentiels d'AppsGate (Section 4) suivis du processus de développement adopté (Section 5) pour aboutir à un déploiement en conditions réelles : dans 5 foyers de personnes externes au projet (Section 6), puis chez les auteurs. De notre expérience « vivre avec » de près de 4 ans, nous sommes en mesure d'énoncer des recommandations à la fois techniques et méthodologiques (Section 7), mais aussi de répondre à la question « un écosystème programmable est-il désirable ? » (Section 8), puis de conclure.

2 METHODES DE CONCEPTION POUR L'HABITAT RESIDENTIEL

L'habitat résidentiel en tant qu'objet de recherche est étudié depuis plus d'un siècle dans des champs disciplinaires aussi variés que l'architecture et les sciences sociales, plus récemment en IHM (Nembrini & Lalanne, 2016) et en informatique. Quels que soient l'objectif et le domaine de recherche, la sphère domestique est un objet dynamique, à facettes multiples, riche en nuances et en exceptions, où s'impose le respect de la sphère privée. Cette complexité a conduit les chercheurs à adapter les méthodes d'analyse des besoins et de conception ou en créer de nouvelles. Certaines sont conduites en laboratoire ou via Internet, d'autres le sont *in situ*.

2.1 Analyse des besoins, méthodes de conception en laboratoire ou via Internet

Les interviews, les ateliers de créativité, les *focus groups* incluant divers supports comme les questionnaires, scénarimages, grilles à remplir (Kahneman et al., 2004) ou jeux, avec ou sans réalisation basse fidélité éventuellement couplée à un Magicien d'Oz, sont les classiques de base. Dans le cas de l'habitat résidentiel, la technique des mimes (*user enactments*) est bien adaptée à l'exploration du futur (Odom et al., 2012). Il s'agit de demander à des utilisateurs représentatifs d'exécuter des scénarios dans un environnement physique et social qui simule des situations familières en utilisant des solutions techniques nouvelles censées répondre à ces situations. En lien avec cette technique, le *speed dating* permet de valider de manière rapide les résultats préliminaires de l'analyse des besoins, en comparant les besoins identifiés par les concepteurs avec ceux d'utilisateurs représentatifs, identifiant les recouvrements et les divergences par des séances de mimes (Davidoff et al., 2007).

Ces techniques peu coûteuses, surtout utiles dans les phases amont du processus de développement, favorisent la génération et la confrontation d'idées tout en respectant la sphère privée. Encore faut-il que les supports soient bien conçus. Par exemple, un scénarimage ne doit mettre en évidence que deux ou trois éléments clefs en cinq ou six vignettes, n'inclure qu'un minimum de texte explicatif et ne pas représenter de personnage sauf lorsque cela est vraiment indispensable (Truong et al., 2006). Ou encore, il convient de proposer des scénarios en opposition (le blanc et le noir), ceci pour favoriser les discussions (Mancini et al., 2010).

Si ces techniques préservent l'espace privé, l'investigateur ne dispose pas du contexte domestique spécifique à chaque participant. Les questionnaires ou scénarios sur mesure (*tailored scenarios*) qui s'ajustent dynamiquement en fonction des réponses aux questions portant sur les situations personnelles du participant « font moins fictifs », mais ne remplacent pas les investigations de terrain (Brown et al., 2015b).

2.2 Investigations de terrain (*Field studies*)

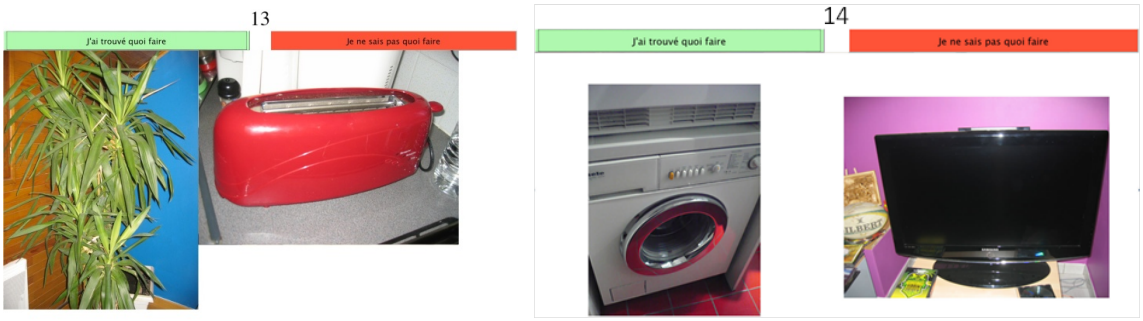
Les interviews et enquêtes de terrain ne permettent pas toujours d'identifier les besoins, préférences et activités routinières. L'habitant n'en est d'ailleurs pas nécessairement conscient, ou bien, décrit ses routines de manière incomplète, volontairement ou pas. La visite vidéo où l'investigateur filme le participant mimant ses activités routinières de pièce en pièce, favorise l'émergence de non-dits (Mitchell et al., 2015). Il n'est cependant pas certain que le participant soit tout à fait sincère. Il est donc nécessaire d'établir une relation de confiance entre le chercheur et la famille, favorisée par des visites préliminaires – incluant par exemple, le partage d'un repas. Ces techniques, qui ont l'avantage de recueillir des données écologiquement valides, se heurtent au problème du respect de la sphère privée.

Les sondes culturelles (*cultural probes*), technologiques et ludiques, qui ne nécessitent pas la présence de l'expérimentateur, ont été introduites comme compromis (Gaver, 1999). Les sondes se présentent sous diverses formes : depuis le journal de bord, le kit incluant un carnet, un appareil photographique et des exercices ou tâches à accomplir, des jeux à jouer en famille (Bernhaupt et al., 2008) jusqu'au dispositif technologique disruptif comme la BinCam (Comber & Thieme, 2013). La BinCam est un prototype de poubelle qui, équipée d'un smartphone, publie sur Facebook des images du contenu de la poubelle, rendant publiques les routines et normes familiales en matière de recyclage des déchets. La disruption comme méthode d'investigation, parce qu'elle provoque, comme la BinCam, un changement brutal fondé sur des idées non conventionnelles, doit cependant être utilisée avec précaution sous peine de mettre en cause la validité des résultats ou de se heurter à des problèmes éthiques (Poole et al., 2015) : le participant peut être mis mal à l'aise en révélant par exemple un comportement socialement déviant ; il peut chercher à plaire à l'expérimentateur, éviter de le décevoir ; il peut aussi se sentir obligé d'obéir aux requêtes de l'expérimentateur ou inversement, oublier d'accomplir des tâches.

Nous avons conçu la méthode DisQo (pour Dispositif du Quotidien) dans le but d'identifier des usages futurs par interconnexion d'objets du quotidien dans l'habitat (Coutaz et al., 2010). Cette méthode d'investigation amont inclut plusieurs outils de mesure visant un bon équilibre entre contrôle expérimental, respect de la sphère privée et validité écologique, le tout en 1h30 par séance auprès de 17 familles de la région grenobloise : prise de photos ; mise en situation ; jeu des associations ; débriefing. L'élément clef de notre méthode est la possibilité donnée aux expérimentateurs de visiter l'habitat par le biais des photos d'objets personnels prises par les participants, puis d'utiliser ces photos comme sonde culturelle ludique (le jeu des associations). En voici l'utilisation :

- Il est demandé à deux volontaires parmi les membres du foyer de prendre chacun dix photos à raison de deux photos par pièce (cuisine, salon, chambre, etc.) – une photo d'objet utile censé simplifier ou organiser les activités domestiques et une photo d'objet superflu dont il serait difficile de se séparer.
- Le jeu des associations a pour objectif de stimuler la créativité des sujets en utilisant comme cartes à jouer, les photos des objets personnels prises par les participants. Comme le montre la Figure 2, un logiciel dédié réalise un tirage aléatoire sur l'ensemble des photos prises par les familles et les présente sur une tablette, deux par deux (couples), puis trois par trois (triplets), avec la consigne d'imaginer des usages futurs si ces objets étaient reliés.

Figure 2. Deux exemples de tirage aléatoire du jeu des associations de la méthode DisQo, les participants devant répondre à la question : « Quel service utile ou superflu apporterait une communication ou coopération entre : - votre plante verte et votre grille-pain ? - votre lave-linge et votre téléviseur ? » [extrait de (Fontaine, 2012)].



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 tout en préservant leur espace privé (les expérimentateurs ne se déplacent pas dans l'habitat) ; les familles s'impliquent dans l'étude et sont intriguées par la suite de l'expérimentation. L'utilisation de photos d'objets personnels dans le jeu des associations, au lieu d'images standard impersonnelles, favorise la créativité, facilite la projection dans des usages futurs limitant ainsi la surcharge cognitive des participants. On trouvera dans (Coutaz et al., 2010) le détail du protocole expérimental ainsi que les résultats utilisés ensuite pour la conception de KISS (*Knit your Ideas into Smart Spaces*), notre tout premier système de programmation domestique par l'utilisateur final (Fontaine, 2012).

2.3 Evaluation

L'Interaction Homme-Machine est aujourd'hui dotée de méthodes éprouvées, qualitatives et quantitatives, portant sur l'évaluation formative et summative de l'utilité et de l'utilisabilité des systèmes interactifs. Encore faut-il, comme le soulignent Greenberg et Buxton, que la méthode choisie soit adaptée aux problème et question de recherche posés (Greenberg & Buxton, 2008). Avec l'émergence de nouveaux champs d'investigation, comme l'Intelligence Ambiante, l'Internet des Objets, l'Interaction Homme-Machine durable (*sustainable HCI*) et les technologies persuasives, les méthodes standard d'évaluation se doivent d'évoluer pour considérer de nouveaux facteurs, au-delà de l'utilisabilité, tels les

facteurs sociétaux et environnementaux, voire la mesure d'impact sur les comportements individuels et collectifs.

Remy et al., dans leur article « *Evaluation beyond usability* … », montrent la difficulté d'évaluer des prototypes conçus pour un futur quelque peu spéculatif : quelle couverture retenir ? Quelle priorité ? Mesurer quoi, comment, et où ? sachant que l'évaluation doit être effectuée au présent pour un contexte futur incertain (Rémy et al., 2018). Dans le cas précis de la programmation de l'habitat par l'utilisateur final, nous estimons que l'expérimentation dans le monde réel et sur la durée doit être un objectif.

2.4 Synthèse

En synthèse, en l'état du savoir-faire méthodologique sur l'habitat domestique, il n'existe pas de méthode universelle, mais une diversité d'approches ainsi que le montrent Brown et al. (Brown et al., 2015a), de même que Desjardin et al. (Desjardin et al., 2015). Il convient donc de choisir les techniques adaptées à la fois aux objectifs de recherche et à l'étape du processus de développement, de les combiner et de les appliquer avec discernement. En particulier, l'évaluation de prototypes et solutions techniques pour l'habitat est peu abordée, notamment en matière de PUF pour l'habitat résidentiel.

3 PROGRAMMATION PAR L'UTILISATEUR FINAL POUR L'HABITAT RESIDENTIEL

Nous distinguons successivement l'offre du marché des résultats de la recherche portant sur la programmation par l'utilisateur final d'habitat résidentiel.

3.1 Offres du marché

Des dizaines de systèmes de domotique programmables par l'habitant sont disponibles sur le marché. ZipaBox[1], Zibase[2], Vera[3], HomeSeer[4] et eeDomus[5] en sont des exemples représentatifs. On trouvera dans (Demeure et al., 2015) une analyse de l'usage de ces solutions, résultat d'une enquête de terrain auprès de 10 foyers de la région grenobloise dont l'un des membres est technophile et amateur de domotique. En synthèse, il s'agit de sécurité (détection d'intrusion et d'inondation), de suivi de la consommation énergétique et des températures, d'automatisation des volets roulants et de la lumière par souci de commodité, et de notifications pour éviter d'oublier des événements de la vie quotidienne (par exemple, la fin de cycle du lave-linge).

Certains systèmes de domotique fournissent des traces sur l'évolution des équipements, mais sans inclure d'aide graphique pour repérer des causalités entre changements d'état. Tous s'appuient sur le paradigme de programmation par règles, mais peu permettent d'abstraire un ensemble de règles en une nouvelle entité – le programme, utilisable à son tour comme unité dans un autre programme.

3.2 En recherche

En recherche, les outils de programmation pour l'habitant ont rarement franchi la preuve de concept et ne couvrent qu'un aspect précis de l'espace problème dont les dimensions

[1] https://www.zipato.com/product/zipabox, dernière consultation : septembre 2018.
[2] https://www.zodianet.com/toolbox-zibase/zibase-classic.html, dernière consultation : septembre 2018.
[3] http://getvera.com, dernière consultation : septembre 2018.
[4] https://homeseer.com, dernière consultation : septembre 2018.
[5] https://www.eedomus.com/fr/, dernière consultation : septembre 2018.

clefs incluent *a minima* : le paradigme de programmation et la syntaxe, la technique d'interaction, et l'aide à la mise au point. Dans ce qui suit, nous présentons les travaux pionniers les plus marquants, puis nous illustrons les dimensions de l'espace problème avec les contributions les plus pertinentes.

3.2.1 Des travaux pionniers preuves de concept

Jigsaw (Humble et al., 2003), CAMP (Truong et al., 2004), iCAP (Dey et al., 2006) et a CAPpella (Dey et al., 2004) comptent parmi les travaux pionniers en PUF pour l'habitat. OSCAR porte sur la composition et le contrôle par l'utilisateur de dispositifs et de services domestiques, par exemple, rediriger sur l'écran de télévision, l'image d'une caméra placée devant la porte d'entrée (Newman et al., 2008). L'utilisateur constitue ainsi des configurations (appelées « setup ») réutilisables dans d'autres configurations. FedNet (Kawsar et al., 2008) explore une nouvelle approche fondée sur l'utilisation de cartes RFID visant à faciliter l'installation d'équipements par l'utilisateur. L'installation (incluant l'appairage) est un problème difficile à résoudre dans toute sa généralité. TeC (Sousa et al., 2011) et Pantagruel-DiaSuite (Drey & Consel, 2012), qui incluent des outils de vérification formelle, sont bien armés pour des déploiements en conditions réelles.

À notre connaissance, à l'exception de Pantagruel-DiaSuite et d'AppsGate – notre proposition présentée en partie 4, ces travaux pionniers n'ont pas fait l'objet de déploiement en conditions réelles et sur la durée.

3.2.2 Paradigmes de programmation et syntaxe

Concernant l'activité de programmation, le paradigme dominant reste la programmation par règles (Ur et al., 2014), à la fois facile d'accès et complexe par manque de structuration. CCBL (Cascading Context Based Language) vise à répondre à cette dualité (Terrier et al., 2017). TeC, de même que ViSiT (Akiki et al., 2017), utilisent les flux de données comme style architectural de composition (Garlan & Shaw, 1994).

Diverses syntaxes concrètes ont été proposées sans qu'aucune ne démontre un avantage déterminant : langages visuels (Pantagruel-Diasuite, TeC), mélange de texte et d'icônes (IFTTT[6]) connexion d'icônes façon puzzle pour ViSiT et Puzzle (Danado & Paternò, 2012), plusieurs adaptations du langage Scratch (Resnik et al., 2009) initialement conçu pour l'initiation à la programmation, mais aussi des langages pseudo-naturels pour CAMP et TARE (Ghianni et al. 2017, Corcella et al. 2017).

La programmation par règles en langage pseudo-naturel est le choix retenu pour AppsGate pour son faible coût d'entrée. Cet aspect sera détaillé dans la Section 4.3.

3.2.3 Techniques d'interaction

Les techniques d'interaction pour construire et éditer un programme utilisent généralement le glisser-déposer entre palettes d'outils et zones d'édition, parfois l'interaction tangible comme pour les Media Cubes (Rode et al., 2005) et les HomeRules (De Russis & Como, 2015), plus rarement la programmation par démonstration (Chin et al., 2006 ; Dey et al., 2006 ; Li et al., 2017a ; Li et al., 2017b).

Concernant l'approche par démonstration, la technique retenue dans Episodite (Li et al., 2017a) est originale, avec toutefois des limites : l'utilisateur construit automatiquement des scripts en agissant directement sur les IHM des Apps de contrôle fournies avec les dispositifs. Par exemple, l'utilisateur qui souhaite établir une lumière douce pour regarder la

[6] https://ifttt.com. Dernière consultation : juillet 2018. IFTTT tient son nom de son paradigme de programmation : « If This Then That ».

télévision dès qu'il entre dans le salon, lance Episodite en mode démonstration, puis utilise l'IHM de contrôle des lampes Philips Hue pour produire l'ambiance lumineuse désirée, puis celle de la télévision pour l'allumage sur la chaîne souhaitée, enfin donne un nom à ce script auquel il associe un capteur de mouvement comme déclencheur d'exécution du script.

L'approche Episodite permet de s'affranchir de l'enfermement technique qu'imposent les solutions SmartThinQ de LG, SmartThings de Samsung, Home Depot de Wink ou encore de WeMo. Si Episodite assure l'interopérabilité entre dispositifs émanant de constructeurs distincts, les Apps dépendent d'Android. De plus, la bonne exécution des scripts exige d'une part, que le téléphone, qui sert de *Hub* à l'habitat, soit connecté aux dispositifs référencés dans les scripts et d'autre part, qu'aucun appel téléphonique ne survienne pendant l'exécution de scripts qui, alors, se termineraient en échec.

Sugilite (Li et al. 2017b), qui relève de la même approche qu'Episodite, est un rare exemple de programmation par démonstration avec interaction multimodale de type « Complémentarité séquentielle » au sens de (Coutaz et al., 1995) : l'utilisateur peut énoncer une commande oralement. Si celle-ci est incomplète, l'utilisateur est invité à désigner (vocalement ou par pointage dans l'IHM graphique des Apps pertinentes) les éléments manquants (par exemple, les paramètres de la commande). En s'appuyant sur la hiérarchie des *widgets* graphiques de l'IHM des Apps sollicitées, Sugilite est capable de généraliser et de générer un script textuel éditable. Contrairement à la technique usuelle des API, l'interopérabilité de Sugilite et d'Episodite repose étroitement sur la connaissance de la structuration interne de l'IHM graphique. Si celle-ci vient à changer, tout comme les API d'ailleurs, le script original devra être modifié.

Dans AppsGate, nous avons retenu une approche conservatrice, mais éprouvée : celle de l'éditeur syntaxique que nous présentons en 4.3.

3.2.4 Aides à la mise au point et dynamicité

À notre connaissance, l'état de l'art sur les PUF/DUF pour l'habitat ne révèle aucune aide à la mise au point. Pareillement, trop peu de solutions, contrairement à AppsGate, gèrent l'apparition et la disparition dynamiques des dispositifs de bout en bout, c'est-à-dire depuis l'infrastructure d'exécution jusqu'à l'IHM. Sur ce point, nous nous sommes appuyés sur les insuffisances rapportées dans la littérature en systèmes distribués, mais aussi sur les méthodes de conception centrées utilisateur portant sur l'habitat résidentiel.

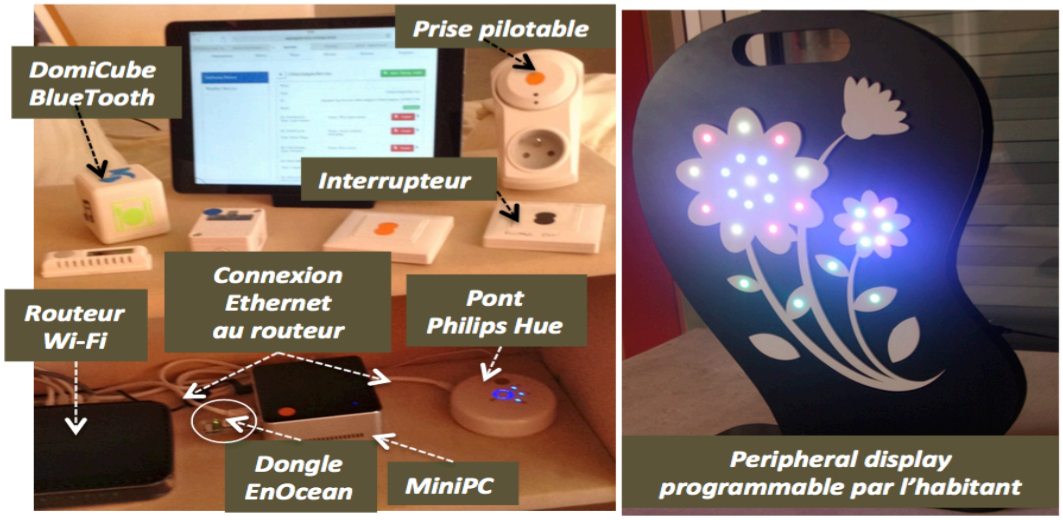
4 APPSGATE : PRINCIPES DIRECTEURS ET LIMITES

AppsGate est l'un des résultats du projet éponyme AppsGate. Le projet AppsGate, Eureka-Catrene CA 110 (2012-2015), avait pour objectif la réalisation de la *set-top box* du futur, plate-forme ouverte et générique, facilitant le développement et le déploiement de capteurs et de services pour l'habitat résidentiel[7].

AppsGate, en tant qu'écosystème programmable, objet de cet article, répond à cinq principes directeurs : (1) Equipement transportable et robuste pour la conduite d'expérimentations *in situ* ; (2) Construction de sens par l'habitant en programmant ; (3) Programmation en langage pseudo-naturel comme compromis entre pouvoir d'expression et simplicité ; (4) Techniques interactives d'observation et d'exploration pour comprendre l'état de l'écosystème ; (5) Support au repérage des anomalies. Un film montrant l'utilisation d'AppsGate est disponible ici[8].

[7] http://iihm.imag.fr/contract/appsgate/
[8] http://iihm.imag.fr/demos/appsgate/appsgate2015.mp4. Depuis le tournage de cette vidéo, SPOK (Simple Programming Kit) a été renommé AppsGate.

Figure 3. Equipement d'AppsGate : (a) un minikit déployable sans dommages pour le lieu d'accueil. (b) Les Fairylights, un cordon de leds programmables, comme afficheur périphérique programmable.



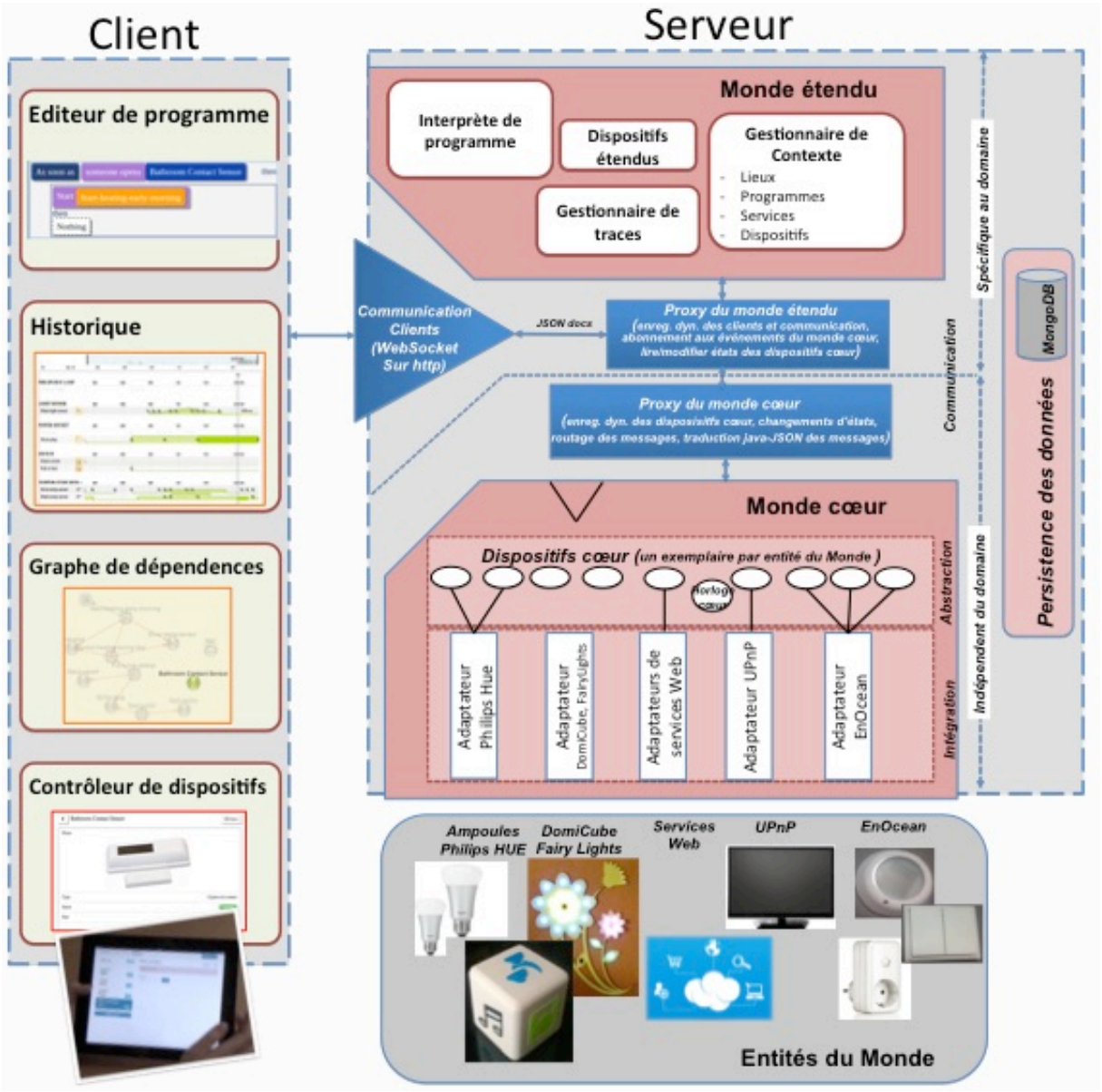
4.1 Équipement transportable et robuste

Comme le montre la Figure 3, AppsGate utilise du matériel transportable et peu encombrant, facilement déplaçable pour être installé dans tout habitat. Il s'agit d'un minikit qui tient dans une valise, comportant un ensemble de capteurs et d'effecteurs sans fil reliés à un Mini-PC via deux sentinelles (*dongles*) branchées sur les ports USB du Mini-PC. L'une sert à la communication avec les capteurs et effecteurs répondant au protocole EnOcean[9], la seconde correspond au protocole BlueTooth. Le mini-PC est à son tour relié à Internet pour l'accès à des services Web. Le DomiCube (en haut à gauche de la Figure 3a) est le résultat d'une réflexion conduite sous forme de *focus group* par des personnes retraitées sur le thème du « contrôle de dispositifs chez soi ». Equipé d'un accéléromètre et d'un magnétomètre, le DomiCube détecte les changements de face et les rotations qu'il communique au Mini-PC via BlueTooth. Le DomiCube, de même que l'afficheur périphérique de la figure 3b sur lequel nous reviendrons dans la section 8, ont été fabriqués dans notre FabLab[10].

Sur le plan logiciel, le cœur fonctionnel de l'écosystème (ou serveur) est exécuté sur le Mini-PC tandis que l'interface homme-machine (IHM), s'exécute simultanément sur autant de machines clientes que souhaitées, principalement des tablettes et des laptops. Pour des raisons de sécurité, toutes les données sont enregistrées sur le mini-PC, non pas sur le *cloud*. La Figure 4 en montre l'architecture logicielle. Le serveur est implémenté en Java et s'appuie sur OSGi (Open Services Gateway initiative) pour gérer l'arrivée et le départ dynamiques des dispositifs et des services Web tiers, en l'occurrence Google Calendar, Google mail, Yahoo ! Weather Forecast, et un service Text-to-Speech. Les clients sont des applications Web implémentées avec HTML5, CSS et JavaScript. La communication entre clients et serveur est assurée par Websocket avec échanges de documents JSON.

[9] https://www.enocean.com, dernière consultation, 26 septembre 2018.
[10] https://amiqual4home.inria.fr/, dernière consultation, 26 septembre 2018.

Figure 4. Représentation de l'architecture logicielle d'AppsGate.



Comme la plupart des plates-formes servant de structure d'exécution à des services pour l'habitat intelligent tels HomeOS (Rosen et al., 2004), OpenHAB[11], FedNet (Kawsar et al., 2008) ou Diasuite (Drey & Consel, 2012), le serveur est structuré en deux niveaux d'abstraction : le « Monde cœur », indépendant du domaine applicatif, et le « Monde étendu » spécifique au domaine applicatif. Le « Monde cœur » est un intergiciel d'intégration des entités du « Monde externe » qui comporte une diversité de services, de protocoles et de dispositifs. L'adaptateur est l'une des solutions usuelles au problème de l'hétérogénéité. Comme le montre la Figure 4, le « Monde cœur » inclut un adaptateur dédié par protocole et des classes de dispositifs abstraits, dits « Dispositifs cœur », servant de représentants normalisés des entités du « Monde externe ».

Les entités du « Monde externe » peuvent être branchées, débranchées – volontairement ou non, ou tomber en panne. AppsGate est robuste à cette volatilité, c'est-à-

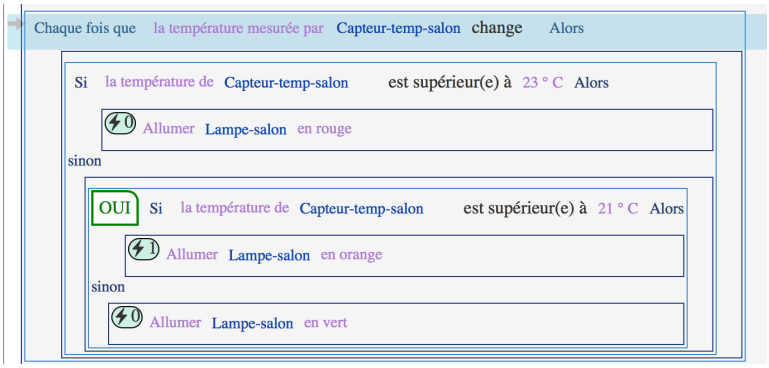
[11] www.openhab.org. Dernière consultation : juillet 2018.

dire résiste à chaud sans avoir à relancer le système : l'arrivée d'un dispositif du « Monde externe » est détectée par le « Monde cœur » qui crée un exemplaire de « Dispositif cœur » qui le représentera et existera jusqu'à la disparition du dispositif externe. Par exemple, lorsque le pont Philips Hue détecte l'arrivée (disparition) d'une ampoule, l'Adaptateur Philips Hue, demande au « Monde cœur » de créer un exemplaire (détruire l'exemplaire) de « Dispositif cœur » qui le représente. Ce changement d'état du « Monde cœur » est enregistré par le « Proxy du Monde cœur » qui, à son tour, diffuse un message JSON à ses abonnés, typiquement le « Proxy du Monde étendu ». Le « Proxy du monde étendu » notifie alors ses abonnés du changement d'état : ses composants et notamment l'interprète des programmes rédigés par les utilisateurs, ainsi que les clients en sorte que leur IHM soit mise en conformité avec l'état de l'habitat.

4.2 Construire du sens en programmant

Les relations entre les entités d'AppsGate, de même que leur comportement, ne sont pas préfabriquées par un spécialiste, ni ajustées par un algorithme d'apprentissage, mais réalisées par l'habitant en programmant. Comme Hinckley (Hinckley, 2017), nous voyons la programmation, non pas comme une affaire de configuration de dispositifs, mais comme une activité créatrice où les personnes définissent leur propre sémantique portant sur l'utilisation des dispositifs et des services de leur habitat.

Figure 5. Le programme *Make-temperature-visible* en cours d'exécution.



Le programme *Make-temperature-visible* de la Figure 5 en montre un exemple : la température du salon mesurée par le capteur de température *Capteur-temp-salon* est traduite en lumière colorée, par exemple rouge si la température est supérieure à 23°C. La *Lampe-salon* sert ainsi d'*afficheur périphérique* qui exprime de manière tangible le niveau de confort thermique. La lampe n'est plus seulement un utilitaire d'éclairage. Elle revêt un nouveau sens et de là, une valeur ajoutée personnelle.

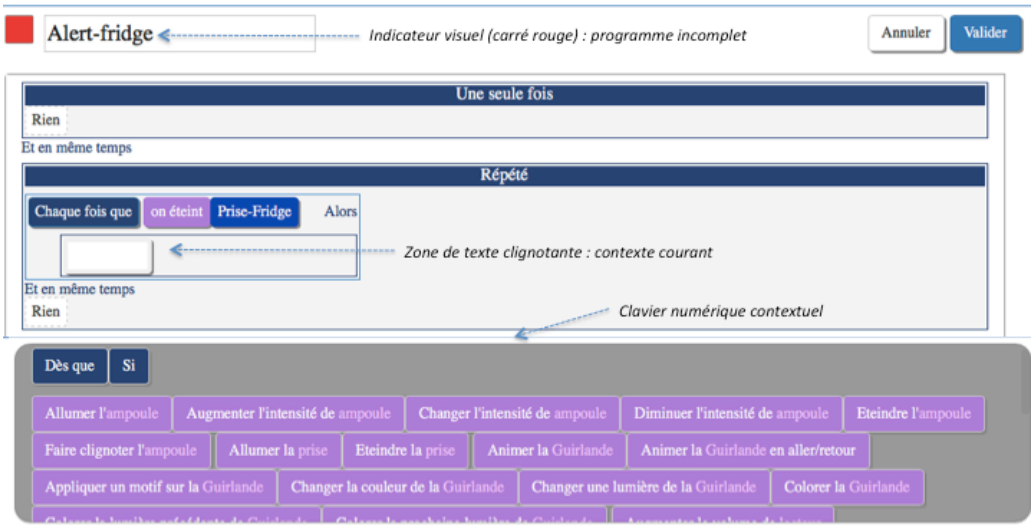
4.3 Programmer en langage pseudo-naturel

La recherche sur la définition de langages de programmation pour l'utilisateur final ne propose pas de solution idéale, seulement des compromis entre facteurs interdépendants, parfois contradictoires. En effet, il s'agit de trouver la meilleure correspondance possible entre la sémantique du domaine propre à l'utilisateur et celle du domaine des langages informatiques.

Nous avons fait le choix d'un langage naturel contraint (anglais ou français) : naturel pour respecter le savoir langagier usuel quotidien, mais contraint pour éviter le traitement complexe des subtilités du langage naturel et notamment les ambiguïtés. On trouvera en Annexe 1 la définition de la grammaire abstraite du langage d'AppsGate. Nous justifions cette définition par un sous-ensemble des facteurs du *Cognitive dimensions Framework* que Green et Petre proposent pour évaluer les langages de programmation (Green & Petre, 1996), notamment le gradient d'abstraction (*abstraction gradient*) et la proximité des

correspondances entre les unités cognitives de l'utilisateur et ceux de la machine (*closeness of mapping*).

Figure 6. Le programme *Alert-fridge* en cours d'édition avec, au bas de la figure, une vue partielle du clavier numérique actualisé pour le point d'insertion courant.



Bon nombre d'études concluent que les habitants expriment facilement et naturellement leurs objectifs sous forme de règles (Ur et al., 2014) et qu'ils les regroupent selon des critères de proximité fonctionnelle (par ex., contrôle de l'éclairage) ou de correspondance avec les activités routinières (par ex., le scénario du lever). Tenant compte de ces études de terrain, nous avons retenu comme fragments : les règles, les conditions et les actions sur les entités du « Monde externe » que connaît l'utilisateur. Ces fragments sont encapsulés sous forme de programmes qui sont alors des abstractions que l'utilisateur définit opportunément par proximité fonctionnelle ou en correspondance avec ses routines.

L'abstraction « programme » constitue un fragment personnalisé que l'utilisateur peut utiliser comme unité dans un programme : un programme peut activer/désactiver un autre programme. AppsGate n'a pas de déclaration explicite de variables : les variables sont les noms que l'utilisateur donne aux entités du « Monde externe » (par ex., *Lampe-salon* de la Figure 5 ou *Prise-Fridge* de la Figure 6). Ces noms sont dynamiquement modifiables et immédiatement réactualisés dans tous les aspects de l'IHM d'AppsGate. Notre choix pour la déclaration implicite de variable est un compromis entre puissance d'expression et *closeness of mapping.*

Un programme se rédige et se modifie au moyen d'un éditeur syntaxique qui minimise les erreurs humaines. Comme l'illustre la Figure 6, dans l'espace d'édition de programme, un clavier numérique de mots et de syntagmes affiché en bas de l'écran contient les options valides pour le point d'insertion actuel dans le programme. Ce guidage qui permet à l'utilisateur de prédire le résultat d'actions futures relève du *feedforward*. On trouvera dans (Vermeulen et al., 2013) une excellente discussion sur les différentes formes de *feedforward* et leur utilisation en interaction homme-machine.

4.4 Observer et explorer pour comprendre

Les programmes en cours d'exécution sont décorés de notations secondaires (au sens de Green et al.) qui permettent de comprendre une partie de l'état de l'écosystème. Dans l'exemple de la Figure 5, l'indicateur *OUI* affiché sur la ligne de l'instruction *Si* pour laquelle la condition vient d'être vérifiée permet de conclure que la température est actuellement comprise entre 21°C et 23°C. Le compteur à la gauche de l'instruction *Allumer Lampe-salon en orange* indique qu'elle a été exécutée une seule fois depuis le lancement du programme. La sélection de ce compteur fait jaillir un message qui indique l'heure précise de cette

exécution. Les autres compteurs du programme valant 0, on en déduit que la température n'a jamais dépassé 23°C, ni n'est passée en-dessous de 21°C durant cette même période.

4.5 Repérer les anomalies

Mais le monde, bien que construit par soi, n'échappe pas aux aléas techniques, ni aux inadvertances, ni aux oublis. Par exemple, une ampoule peut être cassée ou mal vissée sur son support, son alimentation électrique suspendue – il suffit que quelqu'un actionne inopinément l'interrupteur du support.

Figure 7. *Lampe-divan* a disparu : elle est affichée en rouge. L'indicateur d'exécution du programme (triangle en haut à gauche) est orange. Il repassera au vert dès la réapparition de *Lampe-divan*.

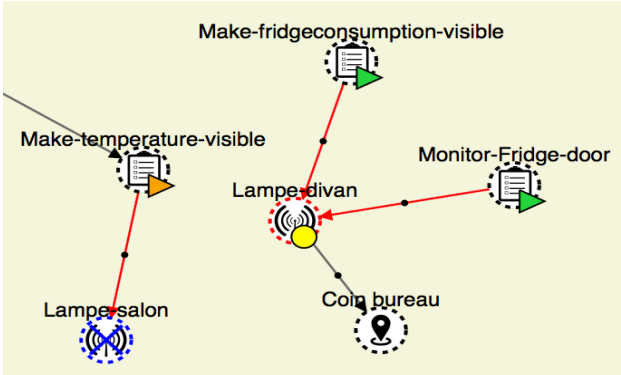
Par inadvertance, l'utilisateur peut modifier l'état d'un même dispositif par plusieurs programmes, par exemple, utiliser la même lampe comme afficheur de température et comme afficheur de consommation d'énergie. Cette concurrence d'accès risque de conduire à des incohérences d'état non souhaitées. Toutes ces conditions sont difficiles à détecter par l'utilisateur, même s'il est un spécialiste. Elles doivent donc être rendues observables.

Figure 8. Extrait du panneau de contrôle des prises pilotables montrant leur état, la consommation électrique instantanée et les actions que l'utilisateur peut leur appliquer.

Les figures 7, 8, 9 montrent comment ce requis d'observabilité est assuré dans AppsGate au moyen de notations secondaires dans les programmes, dans les panneaux de contrôle et dans le graphe des dépendances.

- Tout programme qui ne dispose pas de toutes ses ressources voit son indicateur d'état passer de vert à orange (cf. Figures 7 et 9). De plus, dans le corps de ces programmes, les ressources incriminées sont mises en évidence (*Lampe-divan* de la Figure 7 est affichée en rouge). L'exécution du programme n'est pas interrompue, mais les actions sur ces ressources sont sans effet.
- Un panneau de contrôle permet d'observer et, si besoin, de changer directement l'état des dispositifs (cf. Figure 8). Les ressources inopérantes disparaissent des panneaux de contrôle puis réapparaissent lorsqu'elles reprennent leur fonction.
- Le graphe des dépendances concrétise les relations que l'utilisateur a établies en programmant, relations susceptibles d'être oubliées, le temps passant (cf. Figure 9). Dans ce graphe, les dispositifs inopérants sont représentés sous forme d'un cercle bleu barré d'une croix. Et les dispositifs dont l'état est modifié par plusieurs programmes sont cerclés de rouge.

Figure 9. Extrait du graphe des dépendances, ici filtré pour n'afficher que les lampes et les programmes en cours d'exécution. *Lampe-divan* est actuellement allumée (cercle jaune) mais son état est modifié par 2 programmes en cours d'exécution (cercle en pointillé rouge). *Lampe-salon* n'est pas opérationnelle (croix et cercle bleu). Elle est référencée par le programme *Make-temperature-visible* dont l'indicateur d'exécution (triangle orange) restera orange jusqu'au retour de *Lampe-salon*.



Ayant présenté AppsGate et motivé nos choix de conception, il convient d'en préciser les limites.

4.6 Les limites d'AppsGate

Les limites actuelles d'AppsGate que nous estimons importantes et qui pourraient être levées, portent sur les outils de développement du Génie Logiciel, la puissance d'expression du langage de programmation et la question du passage à l'échelle.

4.6.1 AppsGate et les bonnes pratiques du Génie Logiciel

S'il inclut quelques éléments d'aide à la mise au point des programmes, AppsGate n'est pas un environnement de développement au sens du Génie Logiciel (Burnett & Myers, 2014). En particulier, la gestion des versions n'est pas couverte. Or celle-ci est utile puisque certains programmes adaptés au fil des saisons constituent *de facto* des versions distinctes. De plus, dès l'instant où plusieurs membres de la famille programment l'écosystème, voire partagent ou réutilisent les solutions d'une communauté, la gestion de version devient indispensable.

Concernant la modularité, celle-ci est traitée dans AppsGate avec la notion de programme, un programme pouvant en appeler un autre. Toutefois, l'utilisateur n'est pas conseillé sur la bonne pratique de la modularité ni sur les conventions de nommage. Il doit les découvrir avec l'expérience.

4.6.2 AppsGate et la puissance d'expression de son langage de programmation

Sur ce point, nous justifions nos choix par le principe du faible coût d'entrée (facile à apprendre d'emblée). L'expérience aidant, l'absence de déclaration explicite de variables et d'expressions conditionnelles composées se révèle contraignante, imposant notamment l'écriture d'instructions If imbriquées connues chez les professionnels pour être des sources d'erreur (cf. exemple de la Figure 5). Prenant appui sur la théorie de Byrne et Johnson-Laird sur les modèles mentaux (Byrne & Johnson-Laird, 1992), Metaxas et Markopoulos proposent CoRE, un éditeur syntaxique qui présente en langage pseudo-naturel les expressions booléennes sous forme normale disjonctive (Metaxas & Markopoulos, 2017). Les résultats sont clairement encourageants et devraient servir d'inspiration pour le développement de futurs langages pseudo-naturels de programmation.

L'utilisateur final n'est pas nécessairement un informaticien naïf (Blackwell, 2017). En effet, l'un des auteurs de cet article souhaiterait avoir accès à Python pour modéliser et reconnaître des situations de vie ou des activités humaines afin de programmer des

comportements parfaitement « situés » au sens de Suchman (Suchman, 1987). Toutefois, l'espace de programmation Python ne doit pas s'imposer d'emblée.

Une façon élégante d'allier faible coût d'entrée et puissance d'expression est d'appliquer le principe des *training wheels* (Carroll & Carrithers, 1984). Ce principe s'inspire de la métaphore des stabilisateurs (training wheels) que l'on installe sur les vélos des enfants pour apprendre sans tomber. En IHM, il consiste à présenter à l'utilisateur les fonctions simples (sur le plan cognitif) qui, alors, les utilisera avec succès puis, prenant progressivement confiance avec le système, découvrira à son rythme les fonctions les plus avancées. Ce principe fut élégamment mis en œuvre dans HyperCard, un système hypermédia précurseur du Web, disponible sur les Macintosh dès 1987 et programmable par l'utilisateur final dans un langage de script appelé HyperTalk.

4.6.3 AppsGate et le passage à l'échelle de son IHM

Si AppsGate passe à l'échelle sur le plan technique, il n'en va pas de même sur le plan IHM, notamment pour le clavier contextuel (cf. Figure 6) et le graphe des dépendances (cf. Figure 9).

Le nombre de mots et de syntagmes du clavier contextuel reflète le niveau de développement de l'écosystème. Au bout de 4 ans de développement, le clavier n'est plus directement observable dans la page d'édition, mais navigable (*browsable*) : il implique dorénavant des actions de défilement vertical (*scroll*). Une organisation hiérarchique telle qu'elle est pratiquée dans les téléphones mobiles, est une option envisageable.

Après 4 ans d'utilisation d'AppsGate, nous avons créé de nombreuses relations entre les services, les équipements, les lieux de vie et les programmes. Le graphe des dépendances est devenu illisible. Certes, il est possible d'appliquer des filtres comme dans l'exemple de la Figure 9. Encore faut-il savoir quel filtrage utiliser pour faire apparaître une éventuelle anomalie. D'autres techniques de visualisation doivent être envisagées.

5 PROCESSUS DE DEVELOPPEMENT D'APPSGATE

L'équipe de développement d'AppsGate comprenait 10 personnes de formations complémentaires : 4 chercheurs (1 chercheur en intergiciel, 3 en IHM), 2 spécialistes en facteurs humains et 5 ingénieurs informaticiens. Au lancement du projet, seuls 2 chercheurs étaient compétents dans le domaine PUF avec le développement de KISS (cf. Section 2.2). Cette étude a permis d'identifier des classes de besoins domestiques (monitoring de l'habitat, gestion des stocks, sécurité des personnes et des biens, rappel et pense-bête, aide à l'organisation, bien-être, partage, communication) et de conclure que la composition d'objets du quotidien par l'habitant était susceptible d'améliorer la qualité de vie (Coutaz et al., 2010 ; Fontaine, 2012).

Concernant AppsGate, nous avons appliqué un processus itératif incrémental, autant que possible centré utilisateur, eu égard à la complexité d'implémentation technique de ce type de système. Nous avons procédé en trois phases :

- Une phase de 6 mois d'investigations préliminaires pour le partage d'un objectif commun par les membres d'une équipe de développement multidisciplinaire.
- Une phase d'implémentation et de déploiement agile sur près de 2 ans à raison d'un cycle hebdomadaire pour *sortir progressivement des conditions expérimentales de laboratoire*.
- Une phase d'expérimentation *in vivo* en deux temps que nous détaillons dans les sections 6 et 7.

5.1 Investigations préliminaires de 6 mois

Sur les six premiers mois du projet, nous avons appliqué les techniques d'investigation usuelles : questionnaires en ligne, focus groups, mimes, ateliers de créativité, de sélection

d'idées et de scénarios. Ces activités ont été conduites soit dans des bureaux, soit dans le *Living lab* Habitat Intelligent du laboratoire, soit *chez des membres de l'équipe de développement*. Certaines séances ont fait appel à des utilisateurs représentatifs recrutés sur Internet, d'autres n'ont impliqué que les membres de l'équipe. En parallèle, le système était étudié sous l'angle de la faisabilité technique : choix du matériel, réutilisation de composants logiciels, architecture, définition des structures de données logicielles et leurs relations, définition des requis fonctionnels prioritaires et critères qualité.

Ces investigations centrées sur l'habitat résidentiel se sont révélées onéreuses en temps et en ressources humaines d'autant qu'elles n'ont pas donné lieu à de nouvelles découvertes au regard de l'état de l'art et des résultats de l'étude amont pour KISS (Coutaz et al., 2010 ; Fontaine, 2012). Elles confirment néanmoins la force du nécessaire co-développement. En effet, il n'a pas suffi aux deux chercheurs du projet KISS de présenter les résultats sur les besoins et requis : il était nécessaire que les membres de l'équipe de développement qui n'avaient pas participé à cette étude, *refissent le chemin par eux-mêmes*. Il convenait de sensibiliser l'ensemble de l'équipe aux problèmes de l'habitat programmable et à partager un objectif commun – ce qui est essentiel pour une équipe multidisciplinaire lorsque certains membres sont davantage concernés par les avancées logicielles que par l'intérêt de l'utilisateur final !

L'élément original que nous recommandons dès cette étape est *l'utilisation des habitations de membres de l'équipe comme Living labs*, en l'occurrence 1 maison et 3 appartements pour un total de 6 séances de 3h chacune, impliquant 18 personnes dont 6 membres du projet. Cette utilisation, qui n'avait pas été prévue dans notre plan de développement, visait à répondre à l'absence de convergence dans nos décisions. Non seulement la cohésion du groupe s'en est trouvée renforcée, mais les scénarios d'usage et les idées, ancrés dans du vécu, ont servi de *référence authentique* pour les étapes suivantes du processus de développement.

5.2 Implémentation et déploiement agiles sur près de 2 ans

Il convient d'insister sur la complexité de la mise en œuvre logicielle de systèmes ambiants comme AppsGate : hétérogénéité des protocoles de communication, instabilité des réseaux sans fil et des capteurs/effecteurs, apparition et disparition dynamiques de services et d'objets. Ces difficultés sont amplifiées par la diversité des contextes matériels d'exécution. Par exemple, les murs en béton armé ont un fort impact sur la stabilité des capteurs et effecteurs. En conséquence, *l'expérimentation en situation réelle étant nécessaire, la sortie du laboratoire doit être envisagée par étapes de difficultés croissantes*. Nous avons procédé comme suit :

- Développement logiciel incrémental et itératif sur 18 mois couplé à des tests d'intégration logicielle d'abord en laboratoire (dans nos bureaux), puis déploiement et tests logiciels dans le *Living lab* du laboratoire ; en parallèle, les 4 spécialistes en IHM et facteurs humains effectuaient des évaluations expertes avec des demandes d'ajustement auprès des développeurs logiciels (souvent rétifs aux modifications !).
- Puis, déploiement pendant 4 mois dans 4 appartements de membres du projet. Cette phase a permis de tester la robustesse du logiciel dans des conditions expérimentales distinctes.
- Enfin, installation d'AppsGate pendant 3 semaines chez 5 familles externes au projet lorsque la couverture fonctionnelle, la robustesse et l'utilisabilité du système ont été jugées conformes aux requis prioritaires fixés (par exemple, pour la robustesse, absence de panne technique pendant 1 semaine). En parallèle, de nouveaux composants fonctionnels étaient développés, intégrés et testés selon les étapes devenues habituelles, c'est-à-dire dans nos bureaux, puis en *Living lab*, puis chez nous.

Pourquoi limiter à 3 semaines les expérimentations chez l'habitant ? Si certaines raisons sont conjoncturelles (manque de ressources humaines pour conduire les expérimentations, pour assurer le bon fonctionnement du système et analyser les résultats dans le temps

imparti au projet), *la difficulté fondamentale tient au recrutement de familles sur la durée*. Aussi, pour pallier cette difficulté, deux d'entre nous ont *vécu avec* pendant plus d'un an, mais seul l'un d'entre nous, co-auteur de cet article, maintient assidûment un journal de bord et se sert d'AppsGate depuis octobre 2014 (soit près de 4 ans). Le système est également installé depuis deux ans au domicile de Pierre[12], un collègue retraité, que nous rencontrons de manière informelle trois ou quatre fois par an, l'occasion de « demander des nouvelles d'AppsGate».

Nous résumons ci-dessous les leçons de cette double expérimentation en externe (Section 6), puis en interne chez les auteurs de cet article (Section 7).

6 EXPERIMENTATION EXTERNE *IN SITU*

Nous résumons ici le protocole expérimental et présentons en synthèse les résultats de cette expérimentation.

6.1 Synthèse du protocole expérimental

L'objectif du protocole expérimental appliqué aux foyers externes porte sur l'évaluation de 3 critères :

- Satisfaction et notamment réjouissance-plaisir d'utilisation.
- Utilisabilité en termes de souplesse d'interaction, de perception et de compréhension des concepts clefs du système concernant la programmation et l'état du système avec une attention particulière portée sur la disparition et la réapparition d'équipement et de service.
- Utilité des fonctions du système (graphe des dépendances, historique, test des programmes en mode simulé) et des services tiers intégrés dans AppsGate (météo, agenda, etc.).

La Table 1 montre la distribution des 5 familles participantes. Toutes habitent la région grenobloise, comportent au moins deux adultes en activité professionnelle, sans connaissance en informatique ni en domotique, mais plutôt technophiles (profils identifiés par questionnaire envoyé par email avant l'expérimentation). Chaque famille a reçu un dédommagement de 200 € sous forme de bons d'achat.

Table 1. Les familles participantes de l'expérimentation externe.

Foyer 1	Couple sans enfant, mais femme enceinte	Appartement
Foyer 2	Couple avec petit garçon de six ans	Maison
Foyer 3	Couple sans enfant	Appartement
Foyer 4	Couple avec petite fille de deux ans	Appartement
Foyer 5	Couple sans enfant	Appartement

Les participants ont signé un document de consentement incluant :

- La description du projet et de ses objectifs.
- La nature et le traitement des données enregistrées par AppsGate (consultables sous forme de lignes de vie comme l'illustrent les Figures 11 et 12).

[12] Nom changé intentionnellement.

- L'annonce et l'objet de chacune des 3 visites de notre spécialiste en facteurs humains à savoir, 1h30 chacune avec enregistrement audio des entretiens, et cela le week-end de façon à rencontrer tous les membres de la famille.
- Les précautions d'usage concernant la sphère privée : attitude à observer avec les invités, utilisation du Google agenda d'AppsGate à réserver à la planification des programmes.
- Les contacts 24h/24h en cas de dysfonctionnement.

À chaque visite, l'expérimentateur proposait une tâche à effectuer ainsi qu'un questionnaire de satisfaction à remplir au cours de la semaine à venir.

À la première visite dite de prise en main, chaque famille a reçu l'équipement de la Figure 2, un catalogue papier décrivant les fonctions des capteurs et effecteurs, ainsi qu'un carnet pour noter les bonnes et mauvaises expériences. L'installation par nos soins s'est limitée à vérifier la connectivité avec le réseau. Aucun exemple de programme n'a été fourni, ni de démonstration d'utilisation. L'expérimentateur s'est servi d'images d'écran sur support papier pour poser des questions permettant d'établir l'utilisabilité « à froid » du système : questions portant sur des fonctions que nous savions *a priori* difficiles à comprendre, par exemple, montrant l'écran du graphe des dépendances (cf. Figure 9) : « *Que feriez-vous pour n'afficher que les programmes en cours d'exécution ?* »), ou pour évaluer la prévisibilité, par ex., « *Que va-t-il se passer si l'on sélectionne ce bouton ?* ». Tâche de la semaine 1 : raconter une anecdote d'utilisation positive ou négative et remplir un questionnaire de satisfaction.

Figure 10. Exemples d'objets imaginés au cours d'un week-end de créativité et présentés aux familles : les pantoufles qui éclairent le sol la nuit ; le Tapiston qui informe sur les allers-venues.

Les entretiens de la seconde visite reprennent la séquence de questions de la semaine 1 afin d'évaluer l'évolution du niveau de compréhension des familles, suivie d'une nouvelle séquence portant essentiellement sur les propriétés de flexibilité et d'atteignabilité de l'espace de programmation, par exemple, « *Concernant la programmation, y-a-t-il des choses que vous n'arrivez pas à exprimer ?* ». Tâches de la semaine : (1) remplir le questionnaire de satisfaction (le même que précédemment) ; (2) choisir dans un catalogue de 19 objets futuristes lesquels ils seraient prêts à acheter et (3) imaginer au moins deux objets avec la consigne suivante : « *Choisissez un objet de chez vous qui vous est utile ou pour lequel vous avez de l'attachement. Comme par magie, rendez-le communicant, interactif, intelligent ! Décrivez ces objets dans les fiches ci-dessous* », les fiches proposant un canevas structurant systématique : nom de l'objet, description générale, détail technologique, exemples d'usage. Ces deux dernières tâches ont pour but d'identifier les orientations des besoins vers des objets de communication ou des objets programmables comme nous l'avions fait pour KISS, 5 ans plus tôt avec la méthode DisQo.

Le catalogue des 19 objets proposés est le résultat d'un week-end de créativité de la première phase. Il inclut des objets aujourd'hui disponibles sur le marché (boules lumineuses, miroir intelligent, pommeau de douche ambiant, réfrigérateur montrant l'intérieur sans ouvrir la porte, localisateur d'objet perdu), mais aussi des objets inventés comme ceux

de la Figure 10. Pour chaque objet, le catalogue reprend la structure des fiches des objets à imaginer par les familles. Par exemple : *Tapiston*[13]*, capteur de pression EnOcean et Wifi, objet programmable, scénario d'usage : Gisèle, 25 ans « Le paillasson me permet d'allumer les lumières en fonction de la personne qui rentre en premier. Si c'est moi, le salon et le couloir s'allument. Si c'est mon fils, c'est le couloir et sa chambre. »*

Les entretiens de la troisième semaine ont porté essentiellement sur les capteurs et effecteurs, la nature des programmes, et un débriefing général.

Le corpus anonymisé des données recueillies au cours des entretiens comprend 5 heures d'enregistrement audio des interviews, les notes informelles prises par l'expérimentateur à chaque visite, les plans des habitats avec la localisation des équipements installés, les programmes et les fiches rédigés par les familles. Pour une meilleure garantie d'objectivité, l'analyse du corpus a été assurée par une spécialiste en facteurs humains externe au projet.

6.2 Principaux résultats

La satisfaction et la compréhension du fonctionnement du système se sont améliorées au cours des trois semaines, allant, sur une échelle de Likert à 5 niveaux (Likert, 1932), de 3.2 (std 0.45) la première semaine à 3.5 (std 0.08) en fin d'expérimentation.

Sur les 31 dispositifs du minikit, les familles en ont utilisé 16 en moyenne (std. 3.15), installant 1 à 3 équipements supplémentaires chaque semaine. Les interrupteurs, les capteurs de contact et les ampoules Philips Hue ont rencontré un vif succès. A contrario, les familles ne maîtrisant pas la compréhension de la notion de lumen, l'unité de mesure du flux lumineux, les capteurs de luminosité ont été peu exploités.

Chaque famille a produit 10 programmes en moyenne (std. 2.66) dont 3 à 5 programmes jugés utiles au quotidien. La rédaction de programmes simples (c'est-à-dire ne comportant pas de conditions composées) a été jugée facile par toutes les familles. Programmer est « *fun* » et même engendre un sentiment réconfortant de réussite. Quelques exemples de programme sont fournis en Annexe 2 montrant notamment l'utilisation de l'heure et du service météo pour gérer l'éclairage ou le réveil.

La première semaine, les familles ont réalisé des programmes exploratoires pour « *voir ce que ça fait* » ou « *ce qu'il est possible d'inventer* ». Puis, les familles ont cherché à améliorer leurs activités routinières ou leur confort. Voici les thèmes couverts en fin d'expérimentation :

- Simplification d'utilisation (Coutaz et al., 2010) : remplacement du réveil-alarme, difficile à programmer, par une lampe Philips Hue (1 famille).
- Paix de l'esprit (Brush et al., 2011) : planifier le démarrage de la cafetière automatiquement chaque matin (1 famille).
- Commodité (Brush et al., 2011) : allumer ou éteindre toutes les lumières au moyen d'un seul interrupteur ou de capteurs de contact installés sur les portes, ou encore par une expression combinant le niveau de luminosité et la situation géographique (par exemple, le lever ou le coucher du soleil) (3 familles).
- Confort et détente : ambiance lumineuse pour regarder la télévision (3 familles).
- Regroupement social : clignotement de lampes pour signaler « *l'heure de passer à table* » (1 famille).
- Sécurité et hygiène : utilisation des lumières et/ou courriel pour signaler des conditions ou événements hors de vue (température inappropriée dans la chambre du bébé, chat entrant dans une chambre, ouverture de la porte d'entrée) (1 famille).
- Rendre tangible la consommation d'énergie au moyen de lampes (1 famille).

[13] Le nom "tapiston" vient des participants du week-end de créativité.

Quant aux objets du futur, l'ensemble des foyers aurait acheté un total de 43 objets de notre catalogue, le localisateur d'objet perdu étant le plus désirable (avec 7 achats), suivi par le pommeau de douche (5 achats) et le miroir intelligent (5 achats). Les objets imaginés par les familles relèvent avant tout des tâches ménagères pour gagner du temps : four connecté pilotable à distance pour gagner du temps sur le préchauffage, hotte de cuisine qui se déclenche automatiquement sur détection de fumée ou d'odeurs de cuisson, aspirateur intelligent qui sait quand nettoyer l'appartement. Viennent ensuite la question de la sécurité (serrure connectée se déverrouillant automatiquement en présence des personnes autorisées) et le confort (volets roulants pilotables selon la météo, toilettes à éclairage adapté la nuit, lit intelligent qui sait réveiller le dormeur en douceur puis de manière progressivement insistante jusqu'à se soulever en position assise).

En synthèse, nos familles participantes pensent qu'AppsGate est suffisamment flexible, agrémenté d'expériences plaisantes comme « *notre fille de 2 ans a vite compris comment allumer la lumière en entrant dans sa chambre* ». Le système est jugé utile mais gagnerait à « *intégrer davantage de services Internet* ». Un seul participant a mentionné un comportement inattendu : une lampe éteinte alors qu'elle aurait dû être allumée. Il en a rapidement identifié la cause grâce aux indicateurs d'anomalie (cf. Figures 7, 8, 9) – une personne du foyer avait éteint la lampe au moyen de l'interrupteur du support.

Toutes les familles, qui commençaient au bout de 3 semaines à en apprécier les fonctions, auraient souhaité garder le système. Pour notre part, nous pensons qu'un déploiement sur 3 semaines ne suffit pas pour rencontrer les vraies difficultés ou pour apprécier la co-évolution naturelle et symbiotique habitants-système (Brangier at al. 2009 ; Licklider, 1960). D'où notre décision de *vivre avec* dite *expérience à la première personne.*

7 EXPERIENCE « A LA PREMIERE PERSONNE » ET RECOMMANDATIONS

De leur analyse de plus de 120 articles de recherche en IHM sur l'habitat, Desjardins et ses collègues concluent que *l'expérience à la première personne* selon laquelle le chercheur vit avec sa création est sous-explorée alors qu'elle offre des perspectives inédites (Desjardin et al., 2015). Etant donné le coût de recrutement de familles participantes représentatives qui veulent bien jouer le jeu sur la durée, de même que les ressources humaines nécessaires au suivi expérimental, nous avons jugé opportun de nous engager dans cette voie.

AppsGate est installé en continu depuis l'automne 2014 dans l'appartement des auteurs. L'un est chercheur en IHM et responsable du projet, le second est chercheur en informatique mais externe au projet. Un journal de bord est maintenu avec assiduité. À ce jour, il y est consigné près de 110 notes portant sur des *bugs* techniques dont on n'explique pas la cause mais que l'on sait contourner, des défauts d'IHM, et surtout des anecdotes d'expérience de vie. Nous en tirons des recommandations sur les plans techniques et méthodologiques, mais aussi un regard sur la désirabilité de l'habitat programmable.

7.1 Recommandations techniques

Nos recommandations portent sur le choix de l'intergiciel au-dessus duquel est implémenté le logiciel applicatif, sur la réutilisation de services tiers, et sur le choix matériel des capteurs/effecteurs.

7.1.1 Un intergiciel approprié

Dans un contexte de recherche, « approprié » signifie : (1) un intergiciel ouvert, soutenu par une large communauté en vue de favoriser la pérennité du logiciel applicatif ; (2) un intergiciel capable d'assurer la découverte dynamique des dispositifs et services, avec (re)composition et (re)déploiement à chaud sans intervention humaine de façon à assurer la disponibilité du système en milieu non contrôlé ; (3) un intergiciel à faible coût d'entrée pour

le développeur d'application qui n'est pas nécessairement un spécialiste en systèmes répartis.

OSGi répond aux deux premiers critères, mais s'avère difficile d'accès aux non-spécialistes. OpenHAB masque la complexité d'OSGi, mais dans sa version disponible au début du projet, il ne permettait pas une gestion satisfaisante de la dynamicité. HomeOS est, par construction, un environnement .net alors incompatible avec les plates-formes iOS et Android.

Au vu de notre expérience longitudinale *in situ*, une gestion robuste de la disparition et réapparition dynamiques des dispositifs et des services est un requis de première priorité : l'habitat est un milieu instable, imprévisible, et incrémental. Un capteur ou un effecteur peut être débranché de manière opportuniste, un autre ajouté, déplacé, etc.

En outre, l'intergiciel non seulement doit gérer la dynamicité mais aussi *signaler les disparitions/apparitions de dispositifs et de services aux applications*. Dans le cas contraire, il devient impossible de rendre observables les anomalies de comportement et ceci jusque dans les fondations bien enfouies de l'écosystème. Par exemple, en septembre 2015, une remarque de notre journal de bord indique que « *tous les programmes qui utilisent des services Internet ont leur pastille d'état en orange. Contrairement à toute attente, AppsGate sert à détecter l'instabilité du réseau local probablement due à l'installation du nouveau routeur.* »

7.1.2 De la réutilisation prudente des services Internet et de leur IHM

La réutilisation de services Internet est l'une des clefs du succès de IFTTT[14]. IFTTT est un service web qui permet à l'utilisateur de créer des programmes appelés *applets*[15] reliant des services et des dispositifs provenant de différents fournisseurs et développeurs pour déclencher des automatismes. Par exemple, l'utilisateur rédige une applet qui, détectant les appels téléphoniques effectués avec son SmartPhone (déclencheur spécifié dans la partie THIS d'une règle), enregistre automatiquement les numéros appelés dans son tableur Google (action spécifiée dans la partie THAT de la règle).

Dans AppsGate, nous avons réutilisé Yahoo! Weather Forecast, Gmail, et un générateur de texte vocal, mais sans réutiliser leurs composants IHM (faciles à reprogrammer dans le respect du style visuel d'AppsGate). Par contre, nous avons réutilisé Google Calendar dans sa totalité en sorte que l'habitant planifie/répète l'exécution/arrêt de ses programmes comme il le fait pour ses rendez-vous. En pratique, l'utilisation sur la durée d'AppsGate appelle deux remarques de portée générale :

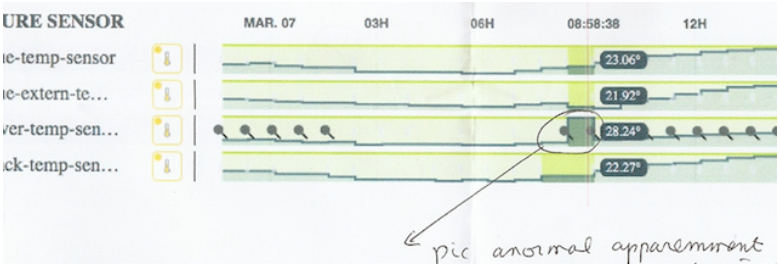
- La réutilisation de services tiers introduit une dépendance technique aux changements de leur API et, par voie de conséquence, nécessite une maintenance logicielle qu'il convient de prévoir dans le cadre d'expérimentations longitudinales. Dans AppsGate, l'API du serveur Text-to-Speech ayant été modifiée, ce service, que nous utilisions pour nous rappeler de « *ne pas oublier le parapluie parce qu'il va pleuvoir aujourd'hui* », n'est hélas plus opérationnel.
- L'IHM d'un service tiers peut introduire des dissonances cognitives plus ou moins marquées. Pour notre part, en tant qu'habitants, nous n'utilisons pas la planification des programmes via Google Calendar, simplement parce qu'il n'est pas notre calendrier usuel et parce que l'accès à l'heure par programme permet d'atteindre un objectif équivalent : « *Chaque fois qu'il est 8h30 démarrer <tel programme>…* ». L'intégration d'un service tiers avec son IHM, doit être soumise au choix explicite ou aux préférences implicites de l'utilisateur. Cette souplesse de choix implique d'être dynamiquement interopérable avec des services dont la nature est non spécifiée à l'avance.

[14] http://ifttt.com, dernière consultation 26 septembre 2018.
[15] Applets, anciennement appelés recettes (*recipes*).

7.1.3 Des capteurs et effecteurs mobiles

Du point de vue de l'expérimentateur, les capteurs et effecteurs sans fil facilitent l'installation et la réutilisation. Nous en avons vu plus haut l'intérêt pour une sortie progressive du laboratoire. Pour l'habitant, cette mobilité signifie liberté et inventivité au gré des idées et des besoins. Notre journal de bord consigne une bonne dizaine de témoignages en ce sens, par exemple : « *j'apprécie de pouvoir mesurer la température là où je suis assise, non pas dans la pièce en général comme le ferait un thermostat*. » ou encore, alors qu'il faisait très froid à Grenoble en février 2017, « *en plaçant des capteurs de température sur les cadres et vitres des fenêtres, nous avons pu comparer sur plusieurs semaines l'efficacité énergétique des ouvrants rénovés avec celle des anciens ouvrants et de là, décider de poursuivre la modernisation du domicile.* »

Figure 11. Pic anormal sur la ligne de vie du capteur de température du salon : il avait été oublié sur la table ensoleillée de la terrasse.



Mais la souplesse que procure la mobilité des capteurs/effecteurs n'est pas sans contrepartie :

- Difficulté à retrouver les capteurs dans l'espace physique. Un visiteur non averti peut les déplacer par inadvertance (telles les petites filles des auteurs qui s'en servent comme jouets !). Ou encore, comme en témoigne l'exemple de la Figure 11, on en oublie la localisation. Il faut alors passer la maison en revue… Un système d'auto-localisation serait donc apprécié.
- Un capteur qui s'éloigne trop de sa sentinelle (ou *dongle* émetteur-récepteur de signaux*),* n'est plus perçu par le serveur et de cela, l'utilisateur n'en est pas averti. La portée d'une sentinelle est impalpable. Elle dépend du type de capteur, de la configuration et de la nature des cloisons. L'enveloppe de bon fonctionnement s'apprend, mais nécessite du temps. Un collègue, directeur de PME dans le domaine énergétique, auquel nous avions prêté un exemplaire d'AppsGate, nous l'a rendu après 1 mois d'essai : « *je ne suis pas allé très loin en raison des murs en béton armé.* »

7.1.4 Des capteurs et effecteurs capables d'introspection

Il convient de choisir des capteurs/effecteurs capables de dévoiler leur fonctionnement interne. Cette remontée d'informations est nécessaire à la fourniture d'explications et de guidage contextuels en sorte que l'utilisateur puisse comprendre, résoudre, voire prévoir les inévitables dysfonctionnements. On trouvera dans (Moussa et al., 2006), l'argumentation pour une démarche d'analyse explicite des dysfonctionnements de systèmes complexes dès les phases amont du processus de conception.

Concernant les capteurs et effecteurs, notre expérience recommande que ceux-ci soient en mesure de fournir dynamiquement les informations suivantes : le niveau de charge, la fréquence d'échantillonnage, leurs états et leurs caractéristiques opérationnelles.

- *Le niveau de charge* afin que le système puisse avertir l'utilisateur que le capteur va bientôt cesser de fonctionner. Les capteurs EnOcean, dotés de cellule photoélectrique, cessent d'émettre après 24h dans l'obscurité. Dans ces conditions, il revient à l'habitant de penser au niveau de charge des capteurs installés dans les lieux plutôt sombres du domicile, un coût attentionnel dont il se passerait volontiers. Le risque est aussi de perdre confiance dans le bon fonctionnement du système. Par exemple, un capteur de

contact censé détecter l'ouverture de la porte d'entrée ne saurait être opérationnel pendant les week-ends, l'appartement étant généralement plongé dans l'obscurité en périodes d'absence.

- *La fréquence d'échantillonnage*, voire la précision, de façon à comprendre si l'absence de remontée d'information correspond à un comportement erratique du capteur (cas fréquent des prises pilotables situées en bordure de l'enveloppe de capture), s'il s'agit d'une période située entre deux prises de mesure, ou si le capteur est défectueux.
- *Capacité du capteur à répondre à une requête sur son état*. À l'exception des prises pilotables, les capteurs EnOcean sont *stateless*, c'est-à-dire ne mémorisent pas leur état. Ainsi, à chaque redémarrage du système, il est nécessaire de forcer un changement d'état des capteurs afin que le système s'initialise en conformité avec l'état du monde physique. Ceci signifie qu' « *à chaque reboot, je dois penser à parcourir tout l'appartement pour ouvrir puis fermer les portes et fenêtres équipées de capteur de contact* ». Cette *statelessness*, nuit *de facto* au passage à l'échelle (grand nombre de capteurs répartis sur plusieurs étages, par exemple).

7.1.5 Des capteurs et effecteurs esthétiques et ergonomiques

L'esthétique, notamment l'encombrement, est aussi un critère de choix. En raison de la taille des prises pilotables, une famille de l'expérimentation externe, tout comme nous, a dû renoncer à certains programmes qui auraient été utiles (voir Figure 3a, en haut à droite, un exemplaire de prise pilotable).

Certains effecteurs, tels les interrupteurs de la Figure 3, présentent de grossiers défauts d'ergonomie. Ces interrupteurs produisent deux types d'événement (appui sur le haut ou le bas du bouton), mais rien ne permet de distinguer visuellement ou tactilement le haut du bas. Nous avons donc décoré chaque interrupteur d'une gommette de couleur comme moyen différenciateur. Par ailleurs, ces interrupteurs reçoivent leur énergie de la force de pression exercée sur le bouton. Si l'on appuie faiblement, le clic-clac du mécanisme physique laisse croire qu'on a produit un événement. L'absence d'effet attendu dans l'environnement physique, par exemple, sur l'éclairage, peut induire de mauvaises hypothèses : *« mon programme doit être faux ou bien AppsGate est en panne. »*

7.2 Cinq recommandations méthodologiques

7.2.1 Ne pas tomber amoureux de sa création

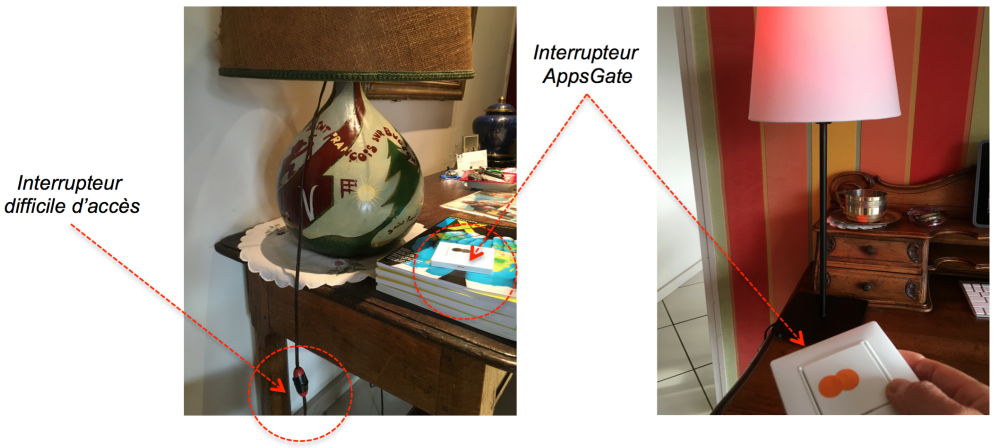
L'expérience à la première personne exige, en tant que concepteur, d'être honnête avec soi-même. En effet, il est tentant de ne pas inscrire dans le journal de bord ce qui est déplaisant, et notamment les *bugs* système ou les erreurs de conception IHM, ou de remettre à plus tard ce que l'on aurait dû noter sur l'instant (croyant ne pas oublier !). Après 6 mois d'utilisation, on peut lire dans le journal : « *Désappairage, sans raison apparente, de tous les dispositifs EnOcean. Je n'ai pas envie de ré-appairer : l'attrait initial du système, probablement dû à la nouveauté, s'estompe.* » : moments de grand découragement, de déception et de frustration qu'il convient de ne pas occulter. Depuis, une restauration du fichier de configuration des dispositifs EnOcean permet de contourner ce problème.

7.2.2 Tenir les rôles conjoints d'habitant naïf et d'expérimentateur critique

Cette dualité d'attitude permet de repérer des phénomènes ou défauts qu'un utilisateur non averti a peu de chance de repérer. Quelques exemples tirés de notre expérience : une note rédigée 2 ans après l'installation du système, remarque que « *nous n'avons pas utilisé le replay*. », service qui permet de rejouer, à la façon d'un film vidéo, les états successifs de l'écosystème et son IHM dans le passé. En conséquence, les 3 hommes-mois dédiés au développement du service auraient été mieux investis dans la programmation par démonstration (ou par l'exemple) comme le démontre cette note du journal illustrée par la Figure 12 : « *Je montre à Madeline* [6 ans] *que l'interrupteur orange allume et éteint la lampe*

ancienne de l'entrée. Sitôt montré, sitôt imité : Madeline prend l'interrupteur, s'approche d'une autre lampe et reproduit mes gestes, hélas sans effet. » Sans expérimentation longitudinale chez nous, ces phénomènes auraient eu peu de chance être d'identifiés comme intéressants.

Figure 12. Programmation par démonstration et proxémie d'équipements. A gauche, par souci de commodité, l'interrupteur difficile d'accès d'une lampe ancienne est complété par un interrupteur d'AppsGate. À droite, l'interrupteur AppsGate est utilisé par Madeline espérant en reproduire l'effet sur une autre lampe par proxémie.



7.2.3 Pousser le système au-delà de son enveloppe fonctionnelle

L'expérience à la première personne doit, si possible, être réalisée chez les membres de l'équipe les moins compétents techniquement : trop averti des limites, l'expérimentateur ne poussera pas le système au-delà de sa couverture fonctionnelle que l'on finit par apprendre implicitement : « *AppsGate marche de mieux en mieux* » nous rapportait notre collègue Pierre après plus d'un an d'utilisation. De même, l'identification d'une limite du langage de programmation (absence de déclaration explicite de variables), est venue du membre de notre foyer qui n'a pas participé au développement du projet. La leçon, *a posteriori*, de cette demande, est qu'il aurait été préférable d'aller au-delà des déclarations implicites permises actuellement et d'appliquer le principe des *training wheels* auquel nous avons fait référence en 4.6.2 (Carroll & Carrithers, 1984).

7.2.4 Tirer avantage des insuffisances du système pour ajuster les protocoles expérimentaux externes

Les lacunes identifiées à la première personne seront nécessairement causes de difficultés pour les familles externes au projet. Si les améliorations souhaitées ne peuvent être apportées au système en temps voulu, le protocole expérimental peut être ajusté et accompagné d'explications pertinentes en sorte que les participants ne soient pas bloqués, risquant d'interrompre l'expérimentation.

Dans le cas du déploiement chez les familles, l'utilisation de nos habitats comme *Living labs* nous a permis de décider d'effectuer l'appairage des équipements avant l'installation chez les familles et d'effectuer nous-mêmes l'installation en sorte que les dispositifs EnOcean soient détectés par la sentinelle branchée sur le Mini-PC. De même, nous avons prévenu les familles de ne pas trop éloigner le DomiCube du Mini-PC en raison de la faible portée de BlueTooth.

7.2.5 Fournir aux participants externes des exemples types de programmes

Notre expérience à la première personne confirme le vécu des 5 familles participantes externes au projet : la valeur ajoutée d'un habitat programmable ne se manifeste pas instantanément. « *Le besoin n'existe pas a priori, il émerge progressivement.* ». Comme pour les familles, notre première semaine ne s'est pas révélée très fructueuse en idées

d'usage. Mais au bout de 2 ou 3 semaines, nos programmes développés se sont révélés très proches de ceux de nos familles (paix de l'esprit, commodité, rendu visible de la température et des consommations énergétiques au moyen de lampes Philips Hue). Il est alors raisonnable de considérer ces programmes comme exemples initiaux à fournir aux participants externes de façon à raccourcir le temps d'appropriation.

À la question sur la désirabilité de l'habitat programmable, notre réponse est : « *oui, mais doit faire beaucoup mieux !* ». Voyons en détail.

8 L'HABITAT PROGRAMMABLE EST-IL DESIRABLE ?

Notre journal de bord témoigne de l'évolution organique de nos relations avec AppsGate. Au bout de 34 mois d'utilisation, une note dit ceci : *nous ne vivons plus « avec le système », nous vivons « dans l'écosystème »*. Des usages inattendus émergent dépassant le pur utilitarisme du tout automatique. Aux idées et nouveaux besoins succèdent aussi des périodes où l'écosystème vit sa vie de manière transparente – *calm technology* au sens de Weiser et Brown (Weiser & Brown, 1995), jusqu'à ce qu'une exception attire l'attention. Par exemple, un blocage-panne est vite repéré : la maison manque de vie. Ce côté *maison animée* a également été relevé par l'une des familles externes.

L'expérimentation *in situ* de l'ordre de l'année pour couvrir l'ensemble des saisons, est incontournable. Notre expérience à la première personne sur près de 4 ans, fait ressortir les thèmes suivants illustratifs de cette nécessité : (1) l'écosystème suit des adaptations dynamiques, (2) un seul capteur suffit pour transgresser la vie privée ; (3) certains besoins sont inspirés par d'autres foyers ; (4) la quantification de l'habitat conduit à l'auto-observation des comportements ; (5) un usage peut en cacher d'autres ; (6) réduire le coût attentionnel reste un défi.

8.1.1 L'écosystème suit des adaptations périodiques

Parmi nos programmes, une bonne vingtaine, certains sont périmés (par exemple, un programme rappelant la prise de médicament à la suite d'une hospitalisation) tandis que d'autres sont activés en fonction des saisons : en décembre, l'éclairage du sapin de Noël doit être contrôlé alors qu'au printemps et en été il convient d'activer le diffuseur anti-moustique si une fenêtre est ouverte en soirée. Enfin, d'autres programmes requièrent quelques ajustements en fonction de l'humeur et des saisons. Par exemple, l'expression tangible de la température ambiante se fait sous forme d'un éclairage rouge chaleureux lorsque, en hiver, elle franchit 23°C indiquant implicitement qu'il est temps de baisser le chauffage (cf. Figure 5), et bleu l'été pour induire une sensation de fraîcheur.

Ces constats d'adaptation périodique sont conformes aux résultats de l'étude ethnographique de Davidoff et de ses collègues (Davidoff et al., 2006).

8.1.2 Un seul capteur suffit pour transgresser la vie privée

La raison d'être des lignes de vie est de permettre à l'habitant d'observer, d'analyser l'évolution de son habitat, de découvrir des corrélations et de là, repérer des routines avec leurs exceptions. En vérité, ces lignes de vie dévoilent le rythme de vie familial bien au-delà de notre intention en tant que concepteurs. Elles permettent notamment de déduire les activités de personnes externes au foyer qui y viennent en votre absence (cf. Figure 13). Depuis ce constat, fort désagréable d'un point de vue éthique, nous avons expliqué le fonctionnement du système à ces visiteurs et nous nous gardons d'inspecter le synoptique correspondant à leurs présences prévues, mais la tentation reste grande !

8.1.3 Certains besoins sont inspirés par d'autres foyers

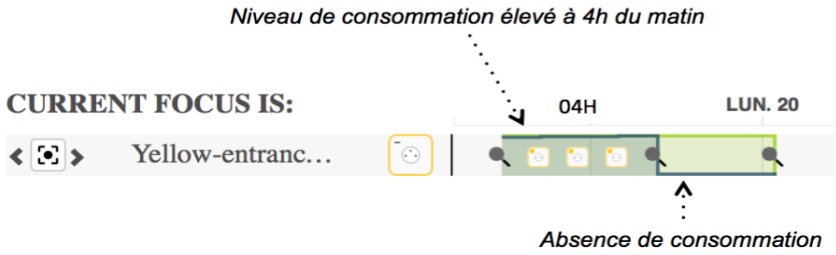
Nos voisins ayant été victimes de vol par effraction, nous avons rédigé un programme qui avertit par courriel qu'une porte vient d'être ouverte en notre absence. Ceci, plus d'un an

après l'installation du système. Dans notre cas, le besoin de sécurité, évident *a posteriori*, ne s'était pas manifesté naturellement alors que ce thème est relevé dans plusieurs études de terrain (Davidoff et al., 2006 ; Coutaz et al., 2010 ; Brush et al. 2011). Le partage de programmes sur les réseaux sociaux mérite donc d'être exploré.

8.1.4 La quantification de l'habitat conduit à l'auto-observation des comportements

Les notes du journal de bord datées du début de l'installation du système font souvent référence à la quantification de notre foyer (*quantified home*). En particulier, nous avons appris le rythme de consommation énergétique de nos appareils ménagers : la bouilloire et le fer à repasser, prétendus « petits » appareils électro-ménagers, sont en vérité très énergivores. Quant au réfrigérateur américain, il se révèle à juste titre le plus gourmand. Dès lors, comment en diminuer la consommation ?

Figure 13. Extrait du synoptique des lignes de vie centré ici sur la prise *Yellow-entrance* située dans l'entrée et reliée à la lampe de la Figure 12. Sa ligne de vie indique une forte consommation électrique vers 4h00 du matin alors que nous étions absents ce lundi 20 : notre petite-fille [20 ans] avait décidé de dormir chez nous en notre absence.



Nous avons procédé à une auto-observation quantifiée de nos comportements en mesurant le nombre d'ouvertures de la porte du réfrigérateur et leur durée, ceci pour une prise de conscience objective de notre comportement susceptible d'être amélioré. Il suffit pour cela d'installer un capteur de contact sur la porte et de rédiger un programme qui repère les événements d'ouverture, envoie un courriel chaque fois que la porte reste ouverte plus de 20 secondes. Nous avons ainsi pris conscience du nombre important d'ouvertures lorsque nous sommes deux à préparer le repas.

L'utilisation d'afficheur périphérique comme les Fairylights programmables de la Figure 3b est aussi un moyen efficace et attractif d'expression en temps réel de la consommation énergétique : nous gagnons une petite lumière LED chaque fois que l'on éteint une lampe reliée à une prise pilotable ou encore nous perdons des fleurs en fonction du budget consommé. En termes de prise de conscience, cette technique a bien fonctionné les premières années d'utilisation d'AppsGate (« *Le réfrigérateur a déjà consommé son budget journalier de 6 kW* »), complétant harmonieusement les mesures en watts (*engineering measures*) des panneaux de contrôle. Mais, le temps passant, l'effet surprise s'estompe, les consommations journalières adoptant un rythme régulier. Aujourd'hui, les *Fairylights* servent de créateur d'ambiance lumineuse adaptée au jour et à la nuit.

8.1.5 Un usage peut en cacher d'autres

Nos notes font état de deux exemples inattendus de détournement de nos usages ou comment (1) le rendu tangible de la température finit par remplir plusieurs fonctions et (2) comment un réfrigérateur devient un agent de communication interpersonnelle.

- *Rendu tangible de la température*. La raison d'être du programme de la Figure 5, rédigé dès l'installation d'AppsGate, est d'exprimer de manière tangible le niveau de confort thermique en allumant *Lampe-salon* avec la couleur idoine. Deux ans plus tard, de manière tout à fait fortuite en un jour frais de fin d'automne, on constate que « *Lampe-salon passe au vert : la température de la pièce a notablement baissé. La pièce est donc suffisamment aérée*. » En somme, cet équipement revêt aujourd'hui différents rôles en

fonction du contexte : détecteur d'aération le matin, détecteur sécuritaire de fenêtre ouverte au moment de quitter l'appartement, afficheur du niveau de confort thermique, et créateur d'ambiance le soir ou au petit matin lorsqu'il fait encore nuit dehors.

- *Le réfrigérateur*. Les messages du réfrigérateur, initialement prévus pour nous faire prendre conscience des ouvertures intempestives de la porte, sont devenus un moyen efficace pour signifier à l'autre membre du foyer qu'il est temps de rentrer : la porte est alors ouverte intentionnellement à d'autres fins que la préparation du repas ! Ces courriels permettent aussi à celui qui est en mission au loin (décalages horaires) de se recaler sur la vie en France. Aussi par l'envoi (trop) fréquent de messages, le réfrigérateur est devenu un « *spammer domestique* » appelant la rédaction de cette note : « *Pour une personnification des équipements de l'habitat : tout équipement devrait avoir une identité Twitter. Ils publieraient tous sur le Home-Twitter-board* ! » De simples utilitaires, les objets communicants se voient attribuer des attributs d'agents communicants.

Mais, ces avantages ont un prix : le coût attentionnel que nous traitons ci-dessous.

8.1.6 Réduire le coût attentionnel est une nécessité et un défi

Programmer est un investissement cognitif que Blackwell mesure en unités de « coût attentionnel » (Blackwell, 2002). En récompense, un programme, s'il est correct, automatise des tâches dont l'utilisateur a actuellement la charge, réduisant ainsi le coût attentionnel futur. En somme, selon Blackwell, l'engagement dans une activité (de programmation ou autre) relève d'une analyse coût/bénéfice/risque : coût de développement, avantages attendus du programme, et risque d'échec. Il convient donc que les solutions techniques en matière de PUF assurent un ratio coût/bénéfice/risque favorable.

Or, dans le cas d'AppsGate, au-delà de la création et maintenance de programmes, l'installation ou la réinstallation d'équipements, l'utilisation des services du système comme la consultation des lignes de vie, sont autant de tâches qui viennent s'ajouter aux activités du quotidien. A cet égard, il est arrivé aux auteurs de remettre à plus tard la mise en œuvre d'une nouvelle idée car utiliser un ordinateur (ou une tablette) représentait, en cette circonstance-là, un effort conatif trop important. Nous proposons quelques pistes en conclusion, et notamment la capacité de déléguer à « la maison » la tâche de programmation.

9 CONCLUSION

En tant que chercheurs-concepteurs de systèmes à usage domestique, il faut oser avoir recours à l'expérimentation personnelle, ceci à toutes les étapes du processus de développement. Les puristes dénonceront la subjectivité de l'approche. De notre point de vue, et en accord avec Desjardin et ses collègues (Desjardin et al., 2015), les expérimentations usuelles avec des participants représentatifs, ne sont pas non plus exemptes d'interprétations subjectives. *A minima*, on peut affirmer que « *si ça ne marche pas pour soi, ça ne marchera pas pour autrui*.»

L'habitat résidentiel est un milieu complexe où même les routines les plus établies sont émaillées d'exceptions. Dans ces conditions, il est illusoire pour un concepteur et réalisateur de système de prévoir les solutions idoines quand bien même il aurait conduit les études de terrain les plus avancées. Notre proposition, avec l'approche PUF/DUF, est de *rendre le pouvoir à l'utilisateur*. Dès lors, au lieu de subir et consommer des solutions inventées par des techniciens, l'utilisateur peut adapter opportunément, improviser, construire du sens et innover. Le *faire par soi-même* (Do-It-Yourself) a toutefois un coût attentionnel probablement disproportionné au regard des services attendus. Trois pistes de recherche complémentaires sont envisageables.

La première approche visant à réduire le coût attentionnel est purement IHM : pouvoir programmer en interaction fluide *as we are*, sans contrainte additionnelle, n'importe où et à tout instant, sous forme de dialogue naturel. L'habitant décrit l'état souhaité, par exemple, « *rappelle-moi de prendre mon passeport avant que je parte demain matin*», et le système génère le programme correspondant. La capacité de migration de tâche (*task migratability*) propriété phare en IHM (Abowd et al., 1992), trouve ici une belle illustration, mais reste un défi scientifique et technique.

Une seconde piste relève de l'apprentissage automatique : le système reconnaît les situations récurrentes émergentes, et adapte le comportement de l'habitat en conséquence. Cependant, Mozer, qui a développé une maison intelligente en y intégrant des techniques d'apprentissage et dans laquelle il a vécu, estime que le système ne disposera jamais de tous les éléments clefs du contexte pour agir en conformité avec l'intention de l'utilisateur (Mozer, 2004). Le thermostat Nest, incapable de reconnaître les exceptions, illustre au niveau d'un seul dispositif, les difficultés scientifique et technique à surmonter (Yang & Newman, 2013). Par ailleurs, déléguer le pouvoir d'adaptation à la machine, c'est aussi perdre le contrôle. Or, plusieurs études ethnographiques concluent sur la nécessité pour l'utilisateur de garder la main (Davidoff et al., 2006 ; Brush et al., 2011).

De notre point de vue, une troisième approche consiste à faire fonctionner en synergie l'apprentissage automatique et le contrôle humain. Mais il convient de veiller à ce que les raisonnements et processus d'apprentissage soient expliqués à l'utilisateur sur demande, ce que les nouvelles techniques d'apprentissage par statistiques (*deep learning*) ne savent pas [encore] faire. Après tout, l'humain doit rester le plus intelligent des deux !

QUE RETENIR DE CETTE RECHERCHE ?

- ✓ **Un ecosystème programmable est désirable au prix d'un coût attentionnel éventuellement élevé.**
- ✓ **Programmer en langage pseudo-naturel avec support prédictif (*feedforward*) est approprié mais il faut pouvoir programmer « comme on est », « quand on a envie ».**
- ✓ **Les usages évoluent : d'abord « maison quantifiée », puis désintérêt (sauf exception), puis effet symbiose, auto-détournement des usages et attachement.**
- ✓ **Les déploiements en conditions réelles sont complexes et chronophages. Sur ce plan, l'intergiciel sous-jacent est clef.**
- ✓ **Le choix des capteurs/effecteurs mérite attention en termes de mobilité, utilisabilité, introspection.**
- ✓ **L'utilisation d'habitats des membres du projet comme *Living labs* est efficace à toutes les étapes du processus de développement.**
- ✓ **Osons avoir recours à l'expérimentation personnelle : le concepteur est aussi un utilisateur !**

REMERCIEMENTS

Ce travail a été soutenu par le projet européen Eureka-Catrene AppsGate CA110 et l'EquipEx AmiQual4Home ANR-11-EQPX-00. Nous tenons à remercier N. Bonnefond, S. Borkowski et R. Pincent d'AmiQual4Home qui ont réalisé les Fairy Lights et le DomiCube, de même toute l'équipe de développement d'AppsGate sans laquelle ce projet n'aurait pu être mené à bien : M. Bidois, S. Caffiau, J.R. Courtois, R. Dautriche, A. Demeure, E. Elias, J. Estublier, T. Flury, C. Gérard, C. Lenoir, J. Nascimento, K. Petoukhov, P. Reignier, C. Roux, G. Vega. Egalement, nos remerciements vont à Emeric Fontaine pour le développement de KISS, notre premier système de programmation par l'utilisateur final pour la maison intelligente dans le cadre de son doctorat et Nadine Mandran pour la mise au point de la méthode DisQo.

ANNEXE 1 : GRAMMAIRE DU LANGAGE

Ci-dessous, la définition de la grammaire abstraite du langage de programmation d'AppsGate respectant la syntaxe BNF (Backus-Naur Form).

-- Un programme est soit un ensemble non vide d'instructions impératives, soit un ensemble non vide de règles, soit un ensemble non vide d'instructions impératives suivi d'un ensemble non vide de règles. « *Une seule fois* SECTION-IMPERATIVE» permet d'exprimer des initialisations. Cette section est exécutée une seule fois au lancement de l'exécution du programme. Les règles sont évaluées en parallèle de manière continue jusqu'à ce que le programme soit désactivée. Un programme peut être désactivé/activé par l'utilisateur via un panneau de contrôle ou par un autre programme. Un programme peut s'auto-désactiver.

PROGRAM ::= *Une seule fois* SECTION-IMPERATIVE {*puis Répété* SECTION-REGLE} |

{*Une seule fois* SECTION-IMPERATIVE *puis*} *Répété* SECTION-REGLE

SECTION-IMPERATIVE ::= INSTR-IMPERATIVE {*puis* SECTION-IMPERATIVE}

SECTION-REGLE ::= INSTR-REGLE {*et en même temps* SECTION-REGLE}

-- Une instruction impérative est soit une action (par ex., Allumer Lampe-salon), ou une instruction conditionnelle (SI ou DES-QUE). SI et DES-QUE permettent de distinguer les conditions portant sur les états (« *Si* la température de … est supérieure à … » des conditions portant sur les événements (« *Dès que* la température de … change … »).

INSTR-IMPERATIVE::= ACTION | SI | DES-QUE

SI ::= *Si* <condition-sur-état> *alors* SECTION-IMPERATIVE {*sinon* SECTION-IMPERATIVE}

DES-QUE ::= *Dès que* <condition-sur-événement> *alors* SECTION-IMPERATIVE

-- Les règles CHAQUE-FOIS-QUE et PENDANT-QUE permettent de distinguer les déclencheurs d'origine événementielle des (*Chaque fois que* la température de … change… ») des déclencheurs fondés sur les états states (*Pendant que* la lampe … est allumée …).

INSTR-REGLE::= CHAQUE-FOIS-QUE | PENDANT-QUE

CHAQUE-FOIS-QUE ::= *Chaque fois que* <condition-sur-événement> *alors* SECTION-IMPERATIVE

PENDANT-QUE::= *Pendant que* <condition-sur-état> *alors maintenir* <état> *ensuite dès que ce n'est plus le cas alors* SECTION-IMPERATIVE

-- La clause *ensuite dès que ce n'est plus le cas alors* a été introduite pour aider les utilisateurs à se souvenir de spécifier le comportement à adopter pour les actions duratives (par ex., *Pendant que* un badge est inséré dans le lecteur de badge *alors maintenir* toutes les lampes allumés *ensuite dès que ce n'est plus le cas* éteindre toutes les lampes).

ACTION ::= <action> <ensemble-choisi-de-dispositifs> | <commande> <nom-de-service> | *Attendre* <nombre-entier-de-secondes> | *Activer* <nom-de-programme> | *Désactiver* <nom-de-programme> | *Désactiver ce programme*

<action> = action permise par <ensemble-choisi-de-dispositifs>

<commande> = commande acceptée par <nom-de-service> disponible dans le système

<ensemble-choisi-de-dispositifs> = *tous les* <type-de-dispositif> *de* <nom-de-lieu> | <nom-de-dispositif> *de* <nom-de-lieu>

<nom-de-lieu> ::= *tous les lieux* | <nom-de-lieu-de-vie>

<condition-sur-état> = une expression booléenne simple dont l'évaluation retourne VRAI ou FAUX. Elle comprend un symbole dénotant un <état>, un opérateur logique simple (=, ≠, <, >), et une valeur appartenant au domaine de valeurs de ce <state>.

<condition-sur-événement> = dénotation d'un événement dont l'évaluation retourne VRAI dès l'occurrence de cet événement.

<événement> = signal instantané détecté ou généré par le système exprimant le changement d'état d'une entité.

<état> = valeur d'un attribut d'une entité.

<name> dans <nom-de-dispositif>, <nom-de-programme>, <nom-de-lieu-de-vie> est une chaîne de caractères spécifiée par l'utilisateur qui dénote une entité connue du système.

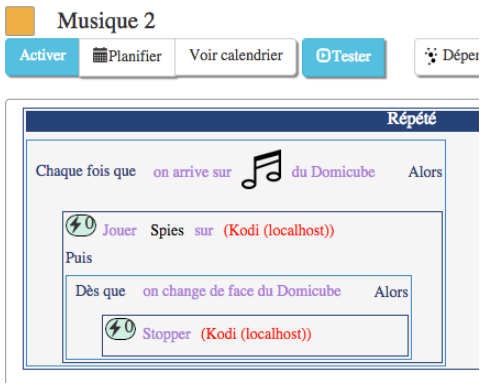
ANNEXE 2 : EXEMPLES DE PROGRAMME

Les programmes ci-dessous ont été rédigés par les familles externes au projet. Ils illustrent la plupart des thèmes observés, de même l'utilisation des service tiers (par ex., Google mail, Yahoo!Weather forecast) intégrés dans AppsGate.

- *Thème : Paix de l'esprit et routine*. Allumer la cafetière, un objet du quotidien utilisé de manière routinière (ici, le matin), branchée sur une prise pilotable (*Prise cafetiere – rouge*) et éteindre cette prise un peu plus tard (probablement par mesure de sécurité).

- *Thème : Commodité et économie d'énergie*. Ne laisser allumé le dressing que si la porte du dressing est ouverte. En effet, la porte fermée ne permet pas de détecter que la pièce est restée allumée inutilement.

- *Thème : confort et détente*. Utilisation du DomiCube pour jouer une pièce musicale sur un lecteur multimédia Kodi, pas (encore) disponible (affichage en rouge).



- *Thème : Routine et organisation sociale* : Il est l'heure du coucher. Utilisation du service intégré Météo Yahoo ! Weather Forecast et de Google mail.

- *Thème : Sécurité*. Utilisation du badge comme dispositif d'indication de présence. Encapsulation de la fonction « détection ouverture porte d'entrée » dans un programme (*détecteur porte d'entrée*) dont il manque une ressource (affichage du nom de ce programme en orange).

- *Thème : sécurité et hygiène*. Ici, avertir via email que la température d'une pièce importante (celle du bébé) est inappropriée.

REFERENCES

- Abowd, G., Coutaz, J., & Nigay, L. (1992). Proc. Of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, Larson & Unger eds, North Holland, 113-130.
- Akiki, P., Bandara, A., & Yu, Y. (2017). Visual simple transformations : empowering end-users to wire internet of things objects, ACM Transaction on Computing Human Interaction (TOCHI), 24(2), article 10.
- Bernhaupt, R., Obrist, M., Weiss, A., Beck, E., & Tscheligi, M. (2008). Trends in the living room and beyond: results from ethnographic studies using creative and playful probing, Computers in Entertainment, 6(1), article 5.
- Blackwell, A. (2002). First Steps in Programming : A Rationale for Attention Investment Model, Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments, 2-10.

- Brangier, E., Dufresne, A., & Hammes-Adelé, S. (2009). Approche symbiotique de la relation humain-technologie : perspectives pour l'ergonomie informatique, Le Travail Humain, vol. 72, Presses Universitaires de France, 333-353.
- Brown, M., Coughlan, T., Ploetz, T., et al. (2015). Editorial : Methods for Studying technology in the Home, Interacting With Computers, special issue on Methods for Studying Technology in the Home, 27(1), Oxford University Press, 1-2.
- Brown, M., Coughlan, T., Blum, J., et al. (2015). Tailored Scenarios: a Low-Cost Online Method to Elicit Perceptions of Home Technologies Using Participant-Specific Contextual information, Interacting With Computers, 27(1), Oxford University Press, 60-71.
- Brush, A. J., Lee, B., Mahajan, R., Agarwal, S., Saroiu, S., & Dixon, C. (2011). Home Automation in the Wild: Challenges and Opportunities, Proc. Int'l Conf. Human-Computer Interaction (CHI 11), ACM New York, 2115-2124.
- Burnett, M., & Myers, B. (2014). Future of End-User Software Engineering: Beyond the Silos, Proc. Int'l Conf. on Software Engineering (ICSE'14), ACM, 201-211.
- Byrne, R., & Johnson-Laird, P.N. (1992).). The spontaneous use of propositionaal connectives, Q.J. Experimental Physchology, 45 (1), 89-110.
- Carroll, J., & Carrithers, C. (1984). Training Wheels in a User Interface, Communication of the ACM, 27(8), 800-806.
- Chin, J., Callaghan, V., & Clarke, G. (2006). An End-User Programming Paradigm for Pervasive Applications, Proc. IEEE Int'l Conf. on Pervasive Services, IEEE, 325-328.
- Comber, R., & Thieme, A. (2013). Designing beyond habit: opening space for improved recycling and food waste behaviors through processes of persuasion, social influence and aversive affect, Personal and Ubiquitous Computing, 17, 1197-1210.
- Corcella, L., Manca, M., & Paternò, F. (2017). Personalizing a student home behaviour. Proc. International Symposium on End User Development (IS-EUD-2017), LNCS 10303, Springer Verlag, 1-16.
- Coutaz, J. & Crowley, J. (2017). Evaluation d'écosystème domestique programmable : Oser « vivre avec » comme méthode expérimentale. Actes 29ème conf. francophone sur l'Interaction Homme-Machine (IHM 2017), ACM, 157-168, https://doi.org/10.1145/3132129.3132138>. <hal-01578502>
- Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., & Young, R. (1995). Four easy pieces for assessing the usability of multimodal interaction: the CARE properties, In Proc. INTERACT'95, IFIP, Lillehammer, 115-121.
- Coutaz, J., & Crowley, J. (2016). A First Person Experience with End-User Development for Smart Homes, In IEEE Pervasive Computing, special issue on Domestic Computing, 15(2), 26-39.
- Coutaz, J., Fontaine, E., Mandran, N., & Demeure, A. (2010). DisQo: A User Needs Analysis Method for Smart Home, Proc. NordiCHI 2010 International Conference, ACM, 615-618.1.
- Danado, J., & Paternò, F. (2014). Puzzle : a mobile application development environment using a jigsaw metaphor. Journal of Visual Languages and Computing, 25(4), 297-315.
- Davidoff, S., Lee, M. K., Zimmerman, J., & Dey, A. (2006). Principles of Smart Home Control, Proc. of the 8th Int'l Conf. on Ubiquitous Computing (UbiComp 2006), Springer-Verlag, LNCS 4206, 19-34.
- Davidoff, S., Lee, M. K., Dey, A., & Zimmerman, J. (2007). Rapidly exploring application design through speed dating, UbiComp 2007: Ubiquitous Computing, Springer Berlin Heidelberg, 429-446.
- Demeure, A., Caffiau, S., Elias, E., & Roux, C. (2015). Building and Using Home Automation Systems: A Field Study, Proc. International Symposium on End-User Development (IS-EUD 2015), Springer, 125-140.

- De Russis, L., & Como, F. (2015). HomeRules : A Tangible End-User Programming Interface for Smart Homes. In Proc. of the 33rd Annual ACM conf. On Human Factors in Computing Systems, Extended Abstract, ACM, 2109-2114.
- Desjardin, A., Wakkary, R., & Odom, W. (2015). Investigating Genres and Perspectives in HCI Research on the Home, Proc. Int'l Conf. Human-Computer Interaction (CHI 15), ACM, 3073-3082.
- Dey, A., Hamid, R., Beckman, C., Li, I., & Hsu, D. (2004). a CAPella : Programming by Demonstration of Context-Aware Applications, in CHI'04 Proc. of the SIGCHI Conf. On Human Factors in Computing Systems, ACM pub., 33-40.
- Dey, A., Sohn, T., Streng, S., & Kodama, J. (2006). iCAP: Interactive Prototyping of Context-aware Applications, Proc. of the 4th Int'l Conf. on Pervasive Computing (Pervasive 2006), Springer, 254-271.
- Drey, Z., & Consel, C. (2012). Taxonomy-Driven Prototyping of Home Automation Applications: a Novice-Programmer Visual Language and its Evaluation, Journal of Visual Languages and Computing, Elsevier, 23(6), 311-326.
- Fontaine, E. (2012). Programmation d'espace intelligent par l'utilisateur final, thèse de doctorat, Uni. Grenoble Alpes, 204 pages.
- Friedman, B., Kahn, P., & Borning, A. (1997). Value Sensitive Design and Information Systems, Human-Computer Interaction in Management Information Systems: Foundations, M.E. Sharpe, 1997.
- Garlan, D., & Shaw, M. (1994). An Introduction to Software Architecture. Carnegie Mellon University, school of Computer Science, report CMU-CS-94-166.
- Gaver, W., Dunne, A., & Pacenti, E. (1999). Design: Cultural probes, Interactions, 6(1), ACM, 21-29.
- Ghianni, G., Manca, M., Paternò, F., & Santoro, C. (2017). Personalization of context-dependent applications through trigger-action rules. ACM Transactions on Computer-Human Interaction (TOCHI), 24(2), article 14.
- Green, T.R.G., & Petre, M. (1996). Usability Analysis of Visual Programming Environments: A Cognitive Dimensions Framework, J. Visual Languages and Programming, 7(2), 131–174.
- Greenberg, S., & Buxton, B. (2008). Usability evaluation considered harmful (some of the time). In proc. of the SIGCHI Conf. On Human Factors in Computing systems (CHI'08), 111-120.
- Hinckley, K. (2017). The editor's spotlight : end-user design for the Internet of Things, a Special issue. ACM Transaction on Computing Human Interaction (TOCHI), 24 :2, paper 8.
- Humble, J., Crabtree, A., Hemmings, T., Akesson, K.P., Koleva, B., Rodden, T., & Hansson, P. (2003). Playing with the Bits, User-configuration of Ubiquitous Domestic Environments, Proc. of the 5th Int'l Conf. on Ubiquitous Computing (UbiComp 2003), Springer Berlin Heidelberg, 256-263.
- Kahneman, D., Krueger, A. B., Schkade, D. A., Schwarz, N., & Stone, A. (2004). A survey method for characterizing daily life experience: The day reconstruction method, Science, 306(5702), 1776-1780.
- Kawsar, F., Nakajima, T., & Fujinami,K. (2008). Deploy Spontaneously: Supporting End-Users in Building and Enhancing Smart Home, Proc. of the 10th Int'l Conf. on Ubiquitous Computing (UbiComp 2008), ACM New York, 282-291.
- Li, T., Li, Y., Chen, F., & Myers, B. (2017a). Programming IoT Devices by Demonstration Using Mobile Apps, In Proc. IS-EUD 2017, 6th International Symposium on End-User Development, Eindhoven, Springer.
- Li, T., Azaria, A., & Myers, B. (2017b). SUGILITE : creating multimodal smartphone automation by demonstration, Proc. Conf. on Human Factors in Computing Systems (CHI'17), ACM, 6038-6049.
- Licklider, J.C.R. (1960). Man-Computer Symbiosis, IRE Transactions on Human Factors in Electronics, HFE-1, 4-11.

- Lieberman, F., Paternò, F., & Wulf, V. (2006). End-User Development, Kluwer Academics.
- Likert, R. (1932). A Technique for the Measurement of Attitudes, Archives of Psychology, vol. 140, 1–55.
- Mancini, C., Rogers, Y., Bandara, A., Jedrzejczyk, L., Joinson, A., Price, B., & Thomas, K., Nuseibeh, B. (2010). Contravision: Exploring Users'reactions to Futuristic Technology, In Proc. of the SIGCHI Conference on Human Factors in computing Systens (CHI'10), ACM New York, 153-162.
- Metaxas, G., & Markopoulos, P. (2017). Natural contextual reasoning for end users, ACM Transactions on Computer-Human Interaction (TOCHI), 24(2), Article 13.
- Mitchell, V., Mackley, K., Pink, S., Escobar-Tello, C., Wilson, G., & Bhamra, T. (2015). Situating Digital Interventions: Mixed Methods for HCI Research in the Home, Interacting With Computers, 27(1), Oxford University Press, 3-12.
- Moussa, F., Kolski, C., Riahi, M. (2006). Analyse des dysfonctionnements des systèmes complexes en amont de la conception des IHM : apports, difficultés, et étude de cas. Revue d'Interaction Homme-Machine (RIHM), 7, 79-111.
- Mozer, M.C. (2004). Lessons from an Adaptive Home, Smart Environments: Technologies, Protocols, and Applications, John Wiley & Sons, Chapter 12.
- Nardi, B. (1993). A Small Matter of Programming; Perspectives an End User Computing, Cambridge MIT Press.
- Nembrini, J., & Lalanne, D. (2016). Quand la Machine devient Bâtiment : Quelle Place en IHM pour l'Interaction Homme-Bâtiment?, alt.IHM, 28ième conf. francophone sur l'Interaction Homme-Machine, Fribourg, 20-33.
- Newman, M.W., Elliott, A., & Smith, T. F. (2008). Providing an Integrated User Experience of Networked Media, Devices, and Services through End-User Composition, Proc. of the 6th Int'l Conf. on Pervasive Computing (Pervasive 2008), Springer, 213-227.
- Odom, W., Zimmerman, J., Davidoff, S., Forlizzi, J., Dey, A. K., & Lee, M. K. (2012). A fieldwork of the future with user enactments, Proc. DIS 2012, ACM, 338-347.
- Poole, E., Comber, R., & Hoonhout, J. (2015). Disruption as a Research Method for Studying Technology in Homes, Interacting with Computers, Oxford University Press, 27(1), 13-20.
- Rémy, C., Bates, O., Dix, A., Thomas, V., Hazas, M., Friday, A., & Huang, E.M. (2018). Evaluation beyond usability : validating sustainable HCI research. In Proc. of the SIGCHI Conf. On Human Factors in Computing systems (CHI'18), paper 216.
- Resnik, M., Maloney, J., Monroy-Hernández, A., et al. (2009). Scratch: Programming for all, Communications of the ACM, 52 (11), 60-67.
- Rode, J.A., Toye, E. F., & Blackwell, A. (2005). The Domestic Economy: A Broader Unit of Analysis for End-User Programming, Proc. Int'l Conf. Human-Computer Interaction (CHI 05), ACM, 1757-1760.
- Rosen, N., Sattar, R., Linderman, R.W., Simha, R., & Narahari, B. (2004). HomeOS: Context-Aware Home Connectivity, Proc. Int'l Conf. on Wireless Networks, CSREA Press, 739-744.
- Schwartz, S.H. (1992). Universals in the content and structure of values: theoretical advances and empirical tests in 20 countries, Advances in Experimental Social Psychology, 15, 1-65.
- Sousa, J. P., Keathley, D., Le, M., Pham, L., Ryan, D., Rohira, S., Tryon, S., & Williamson, S. (2011). TeC: End-User Development of Software Systems for Smart Spaces, Int'l Journal of Space-Based and Situated Computing (IJBSC), 4(1), 257-269.
- Suchman, L. (1987). Plans and situated action : the problem of human-machine communication, Cambridge University Press.
- Terrier, L., Demeure, A., & Caffiau, S. (2017). CCBL: A Language for Better Supporting Context Centered Programming in the Smart Home, Proceedings of the ACM on Human-Computer Interaction – EICS, vol. 1(1), article 14, June 2017.

- Truong, K.N., Huang, E. M., & Abowd, G. (2004). CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home, Proc. of the 6th Int'l Conf. on Ubiquitous Computing (UbiComp 2004), Springer, 143-16.
- Truong, K. N., Hayes, G. R., & Abowd, G. (2006). Storyboarding: an Empirical Determination of Best Practice and Effective Guidelines, Proc. DIS'06, ACM Press, New York, 12-21.
- Ur, B., McManus, E., Pak Yong Ho, M., & Littman, M. (2014). Practical Trigger-Action Programming in the Smart Home, Proc. Int'l Conf. Human-Computer Interaction (CHI 14), ACM, 803-812.
- Vermeulen, J., Luyten, K., Van den Hoven, E., & Coninx, K. (2013). Crossing the Bridge over Norman's Gulf of Execution: Revealing Feedforward's True Identity, Proc. Int'l Conf. Human-Computer Interaction (CHI 13), ACM, 1931-1940.
- Weiser, M., & Brown, J.S. (1995). Designing Calm Technology. Xerox PARC.
- Yang, R., & Newman, M. (2013). Learning from a Learning Thermostat: Lessons for Intelligent Systems for the Home, Proc. of the 2013 Int'l Joint Conf. on Pervasive and Ubiquitous Computing (UbiComp 2013), ACM, 93-102.

BIOGRAPHIE

Joëlle COUTAZ

est professeure émérite de l'Université Grenoble Alpes et membre de l'équipe Ingénierie de l'Interaction Homme-Machine qu'elle a créée en 1990 au LIG. Docteur d'état es Sciences Mathématiques (Grenoble, 1988), membre fondateur de l'AFIHM, elle s'est donné comme objectif la promotion de l'IHM en France. En 2007, elle est nommée docteur *Honoris Causa* de l'université de Glasgow et membre de la SIGCHI Academy de l'ACM. Elle reçoit en 2013 le *Pionneer award* de l'IFIP TC13, et reçoit les enseignes de Chevalier de l'Ordre National de la Légion d'Honneur. Ses sujets de recherche : architecture logicielle des systèmes interactifs (modèle PAC), interaction multimodale, plasticité des IHM, intelligence ambiante et habitat intelligent. En 2008, à la demande du CNRS et du Ministère de l'Enseignement Supérieur et de la Recherche, elle a co-animé avec James Crowley un groupe de travail pour la création en France d'un programme de recherche en Intelligence Ambiante.

James L. CROWLEY

est professeur à Grenoble INP et responsable de l'équipe-projet Pervasive Interaction au LIG et au centre de recherche Inria Grenoble Rhône Alpes. Depuis son doctorat (Carnegie Mellon University, 1981), il a édité 2 livres, cinq éditions spéciales de journaux et publié plus de 250 articles avec des contributions fondamentales en vision par ordinateur et robotique mobile. Il a coordonné plusieurs réseaux et projets européens. Il est nommé à l'Institut de France (IUF) en 2011 et reçoit les enseignes de Chevalier de l'Ordre National du Mérite en 2014. Sa recherche actuelle porte sur la conception de systèmes et de techniques de perception artificielle multimodale pour l'observation des personnes et de leurs activités.