# Évaluation d'Écosystème Domestique Programmable : Oser « Vivre avec » comme Méthode Expérimentale

## *Evaluation of Programmable Domestic Eco-systems: "Living in it" as an Experimental Method*

Joëlle Coutaz
Univ. Grenoble Alpes, CNRS, LIG
F-38000 Grenoble, France
joelle.coutaz@univ-grenoble-alpes.fr

James L. Crowley
Univ. Grenoble Alpes, CNRS, Inria,
Grenoble INP, LIG
F-38000 Grenoble, France
james.crowley@inria.fr

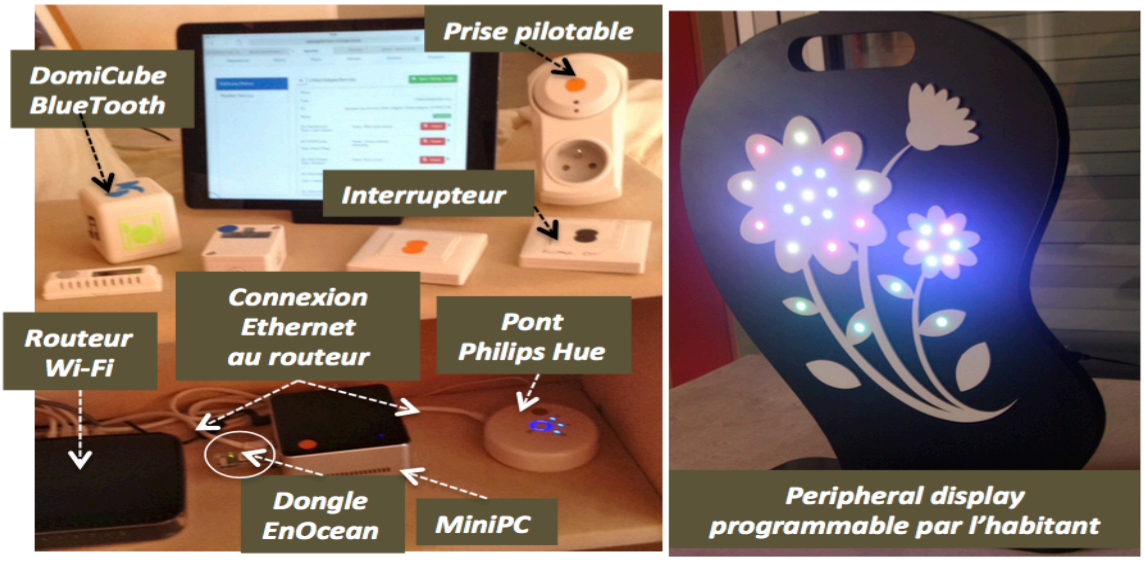


**Figure 1: L'équipement de l'écosystème programmable AppsGate : un minikit déployable sans dommages pour le lieu d'accueil.**

## 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.

## CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction**; HCI design and evaluation methods • **Ubiquitous and mobile computing** → Empirical studies

## KEYWORDS

End-User Programming (EUP), End-User Development (EUD), smart home, experimental method.

## RÉSUMÉ

Cet article présente une 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-CLEFS

Programmation par l'utilisateur final, habitat intelligent, méthode expérimentale.

# 1 INTRODUCTION

La *maison intelligente* fait l'objet d'une attention soutenue en recherche comme en développement industriel. Ce terme couvre toutefois de nombreuses acceptions, depuis l'argument publicitaire et une domotique évoluée fondée sur l'interconnexion d'objets et de services, jusqu'à l'écosystème capable, en toute circonstance, de répondre de manière appropriée aux valeurs de l'habitant.

Or, les valeurs sont volatiles, variant d'une culture à l'autre, entre familles et individus, en fonction des circonstances et à différentes échelles de temps [22, 45]. L'adaptation à la variabilité des valeurs individuelles et collectives constitue un défi scientifique majeur. Nous avons choisi d'explorer la « Programmation par l'Utilisateur Final » (EUP), voire le « Développement par l'Utilisateur final » (EUD), comme un début de réponse à ce défi. Cette approche appelle de nouvelles questions de recherche.

## 1.1 Questions de recherche

L'approche EUP/EUD [7, 31], 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 c'est le cas, un écosystème programmable est-il désirable sachant que la motivation première de l'utilisateur n'est pas d'apprendre à programmer, mais de résoudre un problème personnel qui serait plus coûteux à résoudre sans l'aide du système [36] ? Puisque l'échelle de temps a son importance, les méthodes d'évaluation usuelles de l'Interaction Homme-Machine (IHM) suffisent-elles pour juger de l'adéquation de l'approche EUP/EUD en matière d'écosystème domestique ?

## 1.2 Contributions de l'article

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 EUP/EUD, d'autre part, le processus expérimental.

Concernant l'approche EUP/EUD :

- Façonner son habitat par programmation procure des sentiments réconfortants de réussite en coévolution organique avec la maison qui prend vie sous l'effet de sa propre créativité, 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.

Sur le plan expérimental :

- Les enquêtes de terrain et la panoplie usuelle des techniques centrées utilisateur d'identification des besoins, de conception et d'évaluation en laboratoire sont nécessaires, mais insuffisantes.
- 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 significatif (15 personnes-an pour AppsGate réparties sur près de 3 ans).

Nous proposons donc le compromis suivant : effectuer chez des familles des expérimentations courtes (un mois) complétées par des déploiements longs (une année) 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* ?

Dans ce qui suit, nous proposons une courte analyse des méthodes de conception pour l'habitat domestique avant de décrire les éléments fonctionnels essentiels d'AppsGate. Nous en détaillons ensuite le processus de conception et d'évaluation. On trouvera ici [1] une vidéo montrant les possibilités du système.

# 2 METHODES DE CONCEPTION POUR L'HABITAT DOMESTIQUE

L'habitat domestique 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 [37] et en informatique. Quels que soient l'objectif et le domaine de recherche, la sphère domestique est réputée être un objet dynamique, à facettes multiples, riche en nuances et en exceptions, où le respect de la vie privée est prégnant. Cette complexité a conduit les chercheurs à adapter les méthodes d'analyse des besoins et de conception existantes 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 et *speed dating* [14], grilles à remplir [27], jeux, ou mimes [39] avec ou sans réalisation basse fidélité éventuellement couplée à un Magicien d'Oz – sont les classiques de base réalisables en laboratoire ou en ligne sur Internet.

Ces techniques de faible coût, 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 5 ou 6 vignettes, n'inclure qu'un minimum de texte explicatif et ne pas représenter de personnage sauf lorsque cela est vraiment indispensable [49]. Ou encore, il convient de proposer des scénarios en couple alternatif (le blanc et le noir), ceci pour favoriser les discussions [33].

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

[1] http://iihm.imag.fr/demos/appsgate/appsgate2015.mp4

personnelles du participant « font moins fictifs », mais ne remplacent pas les investigations *in situ* [5].

## 2.2 Investigations de terrain (*Field Studies*)

Interviews et enquêtes de terrain ne permettent pas toujours d'identifier les besoins, préférences et actions 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 [34]. 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 vie privée. Les sondes culturelles (*cultural probes*), technologiques et ludiques, qui ne nécessitent pas la présence de l'expérimentateur, ont été introduites comme compromis [23].

Les sondes se présentent sous de nombreuses variations : depuis le journal de bord, le kit incluant un carnet, un appareil photographique et des « exercices » à faire, des jeux à jouer en famille [1] jusqu'au dispositif technologique disruptif comme la BitCam [10] censée modifier les routines et normes familiales et de là, les révéler sous un angle nouveau. La disruption comme méthode d'investigation doit cependant être utilisée avec précaution sous peine de mettre en cause la validité des résultats ou de se heurter à des problèmes éthiques [41] : 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.

## 2.3 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. [4], de même Desjardin et al. [17]. 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 afin d'éviter les biais. C'est ce que nous avons pratiqué au mieux de nos compétences dans le développement d'AppsGate.

# 3 APPSGATE : PRINCIPES DIRECTEURS

AppsGate répond à cinq principes directeurs : (1) un équipement portable et fiable pour la conduite d'expérimentations *in situ* ; (2) Construction de sens par l'utilisateur 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. Nous terminons cette section en comparant AppsGate à des systèmes similaires représentatifs de l'état de l'art.

## 3.1 Un équipement portable

La Fig. 1 illustre la dimension matérielle d'AppsGate : un ensemble de capteurs et d'effecteurs sans fil communiquant avec 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, la seconde correspond au protocole BlueTooth. Le mini-PC est à son tour relié à Internet pour l'accès à des services Web (Google Mail, Google Agenda, Yahoo!Weather Forecast).

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 lap-tops. Toutes les données sont enregistrées sur le mini-PC (pas de cloud !). Une description technique plus détaillée d'AppsGate est présentée dans [11].

## 3.2 Construire du sens en programmant

Les relations entre les entités d'AppsGate, de même leur comportement, ne sont pas préfabriquées par un spécialiste, ni ajustées par un algorithme d'apprentissage, mais réalisées par l'habitant en programmant. Le programme *Make-temperature-visible* de la Fig. 2 en montre un exemple: associée à un capteur de température, une ampoule Hue conçue par Philips pour créer des ambiances lumineuses, *est transformée par l'habitant en afficheur périphérique* pour traduire le niveau de confort thermique.

## 3.3 Programmer en langage pseudo-naturel

Un programme AppsGate est exprimé en langage naturel contraint (anglais ou français) : naturel pour respecter le savoir usuel quotidien, mais contraint pour éviter le traitement complexe des subtilités du langage naturel et notamment les ambiguïtés. On trouvera dans [11], la définition de la grammaire abstraite du langage ainsi que la justification de nos choix fondés sur le *Cognitive Dimensions Framework* de Green et Petre [24].

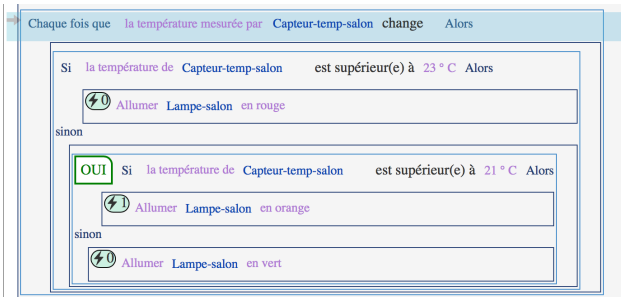


**Figure 2: *Make-temperature-visible* en cours d'exécution.**

Un programme se rédige et se modifie au moyen d'un éditeur, sans risque de commettre des erreurs : en mode « édition de programme », un clavier numérique de mots et de syntagmes affiché en bas de l'écran contient les options qui font sens pour le point d'insertion actuel dans le programme. Ce guidage sur la suite des actions possibles relève du principe de *feedforward* [51].

### 3.4 Observer et explorer pour comprendre

Les programmes en cours d'exécution sont décorés de notations qui permettent de comprendre une partie de l'état de l'écosystème. Dans l'exemple de la Fig. 2, 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 23°C et 21°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.

### 3.5 Repérer les anomalies

Mais ce monde, bien que construit par soi, n'échappe pas aux aléas techniques, ni aux inadvertances. 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 3: *Lampe-divan*, affichée en rouge, a disparu. L'indicateur d'exécution du programme *Make-fridge-consump-visible* (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 4: Extrait du panneau de contrôle des prises pilotables montrant leur état, la consommation instantanée et les actions utilisateur possibles.**

Les figures 3, 4, 5 montrent comment ce requis d'observabilité est assuré dans AppsGate : indicateurs 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. Fig. 3). De plus, dans le corps de ces programmes, les ressources incriminées sont mises en évidence. 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. Fig. 4). 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. Fig. 5). 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.

### 3.6 AppsGate et l'état de l'art

*3.6.1 Offres du marché.* Des dizaines de systèmes de domotique programmables par l'habitant sont disponibles sur le marché. ZipaBox, Zibase, Vera, HomeSeer et eeDomus en sont des exemples représentatifs. On trouvera dans [15] une analyse de l'usage de ces solutions dont la couverture fonctionnelle se rapproche de notre proposition.

Comme AppsGate, certaines fournissent des traces sur l'évolution des équipements, mais sans inclure d'aide graphique pour repérer des causalités entre changements d'état. Toutes 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 dans un autre programme. Typiquement, les programmes *Démarrer-maison* et *Arrêter-maison* encapsulent respectivement une liste d'instructions de la forme *Démarrer <nom de programme>*, *Arrêter <nom de programme>*.

*2.6.2 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, typiquement, le paradigme de programmation et les notations syntaxiques. OSCAR [38], Jigsaw [25], CAMP [48], iCAP [19] et a CAPpella [18] comptent parmi les travaux pionniers en EUP pour l'habitat. Plus récent, FedNet [28] 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. Il a été sous-estimé dans AppsGate. TeC [46] et Pantagruel-DiaSuite [20], qui incluent des outils de vérification formelle, sont bien armés pour des déploiements en conditions réelles.

Concernant l'activité de programmation proprement dite, le paradigme dominant reste la programmation par règles [50], à la fois facile d'accès et complexe par manque de structuration. CCBL (Cascading Context Based Language) vise à répondre à cette dualité [47]. Diverses syntaxes concrètes ont été proposées sans qu'aucune ne démontre un avantage déterminant : langages visuels (Pantagruel et TeC), mélange de texte et d'icônes (IFTTT [26]), adaptations du langage Scratch

initialement conçu pour l'initiation à la programmation [42], mais aussi, comme pour AppsGate, langages pseudo-naturels (CAMP). Le pseudo-naturel fait démodé si l'on s'en tient à l'aspect « com » du système. En pratique, notre expérience montre qu'il présente un faible coût d'entrée.

Les techniques d'interaction pour construire un programme utilisent généralement le glisser-déposer entre palettes d'outils et zones d'édition, parfois l'interaction tangible – par exemple, les Media Cubes [43] ou les HomeRules [16], voire la programmation par démonstration [9, 18, 29].

Concernant l'approche par démonstration, la technique retenue dans EPISODITE [29] 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 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.

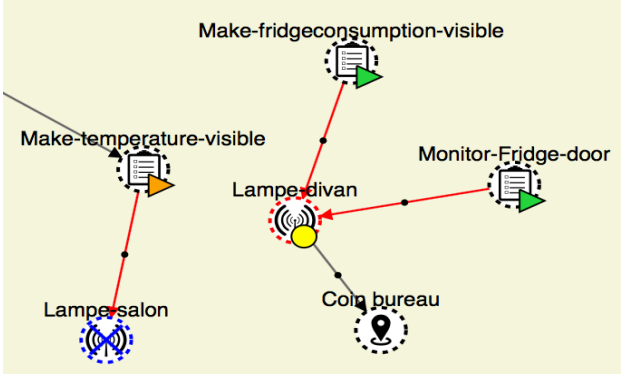


**Figure 5: Extrait du graphe des dépendances, ici filtré pour n'afficher que les programmes en cours d'exécution et les lampes. L'état de *Lampe-divan* est modifié par 2 programmes en cours d'exécution tandis que *Lampe-salon* référencée par *Make-temperature-visible* n'est pas opérationnelle.**

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 doivent néanmoins être exécutables sur 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.

*2.6.3 Aide à la mise au point et dynamicité.* À notre connaissance, l'état de l'art sur les EUP/EUD pour l'habitat ne révèle aucune aide à la mise au point. De même, 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, mais aussi sur les méthodes de conception centrées utilisateur portant sur l'habitat domestique.

## 4 PROCESSUS DE DÉVELOPPEMENT 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 EUP/EUD pour l'habitat grâce à une étude conduite en amont du projet. Cette étude a permis d'identifier des classes de besoins domestiques (observation 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 est susceptible d'améliorer la qualité de vie [12, 21].

Concernant le projet proprement dit, 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. Le développement s'est opéré en trois grandes 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 (cf. 4.1).
- 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* (cf. 4.2).
- Une phase d'expérimentation *in vivo* en deux temps que nous détaillons dans les sections 5 et 6.

### 4.1 Investigations préliminaires de 6 mois

Durant les six premiers mois du projet, nous avons appliqué les techniques d'investigation usuelles : questionnaires en ligne, ateliers de créativité, focus groups, mimes, séances de génération, 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, ontologie logicielle, définition des requis fonctionnels prioritaires et critères qualité.

Ces investigations centrées sur l'habitat domestique 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 [12, 21]. Elles confirment néanmoins la force du nécessaire co-développement. En effet, il n'a pas suffi aux deux chercheurs ayant mené l'étude amont de présenter les résultats sur les requis : il était nécessaire que les membres de l'équipe de développement qui n'avaient pas participé à cette étude, *refassent 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 d'actions, 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.

### 4.2 Implémentation et déploiement agiles sur près de 2 ans

On n'insistera jamais assez 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 se sont trouvées 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 (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 près de 3 ans. Le système est également installé depuis deux ans au domicile de Pierre[2], 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 5), puis en interne (section 6).

## 5 EXPÉRIMENTATION EXTERNE *IN SITU*

### 5.1 Synthèse du protocole expérimental

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

Les participants ont signé un document de consentement incluant : (1) la description du projet et de ses objectifs, (2) la nature et le traitement des données enregistrées par AppsGate (consultables sous forme de lignes de vie – cf. Fig. 6), (3) 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, (4) les précautions d'usage en terme de vie privée : attitude à observer avec les invités, utilisation du Google agenda d'AppsGate à réserver à la planification des programmes, (5) les contacts 24h/24h en cas de dysfonctionnement. À chaque visite, l'expérimentateur proposait une tâche à faire ainsi qu'un questionnaire de satisfaction (questions ouvertes) à remplir pendant la semaine à venir.

À la première visite dite de prise en main, chaque famille a reçu l'équipement de la Fig. 1, 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 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 de dépendances (cf. Fig. 5) : « 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.

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, suivis 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 : remplir le questionnaire de satisfaction (le même que précédemment) et choisir dans un catalogue d'objets

[2] Nom changé intentionnellement.

connectés futuristes (résultat d'un week-end de créativité de la première phase) lesquels ils seraient prêts à acheter (afin d'identifier les orientations des besoins vers des objets de communication ou des objets programmables). Les entretiens de la troisième semaine ont porté essentiellement sur les capteurs et effecteurs, la nature des programmes, et un debriefing 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 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.

## 5.2 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 [32], 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 capteurs de luminosité se sont révélés les plus problématiques en raison du concept de lumen alors que les interrupteurs, les capteurs de contact et les ampoules Philips Hue ont rencontré le plus grand succès.

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.

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 [12] : remplacement du réveil-alarme, difficile à programmer, par une lampe Philips Hue (1 famille).
- Paix de l'esprit [6] : planifier le démarrage de la cafetière automatiquement chaque matin (1 famille).
- Commodité [6] : allumer ou éteindre toutes les lumières au moyen d'un seul interrupteur ou de capteur de contact installé sur les portes, ou encore par une expression combinant niveau de luminosité et situation géographique (par exemple, le lever ou le coucher du soleil) (3 familles).
- Confort et détente : ambiance lumineuse pour regarder la télévision.
- Regroupement social : clignotement de lampes pour signaler « *l'heure de passer à table* » (1 famille).
- Sécurité et hygiène : utilisation des lumières ou email 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).
- Rendre tangible la consommation d'énergie avec des lampes (1 famille).

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. Fig. 3, 4, 5) – 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 [3, 30]. D'où notre décision de *vivre avec* dite *expérience à la première personne.*

# 6 EXPERIENCE "À LA PREMIÈRE PERSONNE"

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 [17]. Etant donné le coût de recrutement de familles participantes représentatives qui veulent bien jouer le jeu sur la durée, de même les ressources humaines nécessaires au suivi expérimental, nous avons jugé opportun de nous engager dans cette voie.

AppsGate est installé en continu depuis 34 mois dans l'appartement des auteurs de cet article. 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. En ce jour, il y est consigné 101 notes portant sur des *bugs* techniques dont on n'explique pas la cause, 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.

## 6.1 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.

*6.1.1 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 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 [40] masque la complexité d'OSGi, mais dans sa version disponible au début du projet, ne permettait pas une gestion satisfaisante de la

dynamicité. HomeOS [44] 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 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.*»

*6.1.2 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. 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 vocal ayant été modifiée à notre insu, 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 préférences implicites de l'utilisateur.

*6.1.3 Des capteurs et effecteurs mobiles.* Du point de vue de l'expérimentateur, les capteurs et effecteurs sans fil en 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.*»

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

- Difficulté à retrouver les capteurs dans l'espace physique. Le temps passant, on en oublie la localisation. Ou bien, un visiteur non averti peut les déplacer par inadvertance (telles nos petites filles qui s'en servent comme jouets !). Il faut alors passer la maison en revue… Un système d'auto-localisation serait donc apprécié.
- Un capteur qui s'éloigne trop de son *dongle*, n'est plus perçu par le serveur et de cela, l'utilisateur n'en est pas averti. La portée du *dongle* 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.* »

*6.1.4 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 et résoudre les inévitables dysfonctionnements et que nous, concepteurs, puissions satisfaire les principes du *design for breakdown* [52]. Notre expérience recommande les informations suivantes dont l'absence risque de mettre en cause ou de fausser les résultats d'une expérimentation longitudinale chez des utilisateurs non avertis :

- *Le niveau de charge* afin que le système puisse avertir l'utilisateur que le capteur va bientôt cesser de fonctionner. Rappelons que 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, une charge cognitive dont il se passerait volontiers.
- *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*. 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).

*6.1.5 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 Fig. 1, en haut à droite, un exemple de prise pilotable).

Certains effecteurs, tels les interrupteurs de la Fig. 1, 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. »*

## 6.2 Cinq recommandations méthodologiques

*6.2.1 Ne pas tomber amoureux de sa création.* L'expérience à la première personne exige d'être honnête avec soi-même. Il est tentant, en effet, 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 notre 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 !

*6.2.2 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 très récente du journal : « *Je montre à Madeline* [6 ans] *que l'interrupteur orange allume et éteint la vieille lampe 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 n'auraient pu être relevés.

6.2.3 *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* de Caroll [8].

6.2.4 *Tirer avantage des insuffisances du système pour ajuster les protocoles expérimentaux externes.* Les manquements identifiés à la première personne seront nécessairement causes de difficultés chez 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.

*6.2.5 Fournir aux participants externes des exemples types de programme.* 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 progressivemen*t. 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.

## 6.3 L'habitat programmable est-il désirable ?

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

Notre journal de bord témoigne de l'évolution organique de nos relations avec AppsGate. Au bout de 34 mois d'utilisation, *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*) 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.)

Il faut néanmoins laisser du temps au temps : l'expérimentation *in situ* de l'ordre de l'année, est incontournable. Voici quelques exemples illustratifs.

*6.3.1 L'écosystème subit des adaptations périodiques.* Parmi nos programmes (une bonne vingtaine), certains sont périmés (programme rappelant la prise de médicament suite à une hospitalisation), d'autres ne sont pas activés mais seront à nouveau d'actualité ultérieurement (contrôle de l'éclairage du sapin de Noël en fonction de l'heure), d'autres enfin requièrent des ajustements relativement fréquents (l'ambiance lumineuse souhaitée varie avec les saisons et l'humeur). Ces constats sont conformes à l'étude ethnographique de Davidoff [13].

*6.3.2 Certains besoins sont inspirés par d'autres foyers.* Nos voisins ayant été victimes de vol par effraction, nous avons rédigé un programme qui envoie un e-mail chaque fois qu'une porte est ouverte en notre absence. Ceci, plus d'un an après l'installation du système. Le besoin de sécurité, évident *a posteriori*, ne s'était pas manifesté de manière intrinsèque au

foyer. *Le partage de programmes sur les réseaux sociaux mérite donc d'être exploré.*

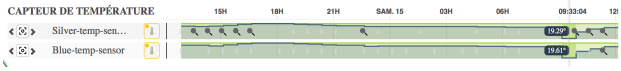


**Figure 6: Extrait du synoptique des lignes de vie centré ici sur deux capteurs de température. La chute brutale des températures vers 9h30 correspond à l'ouverture des fenêtres (ouverture déductible de la ligne de vie de leur capteur de contact).**

*6.3.3 Un seul capteur suffit à transgresser la vie privée.* La raison d'être des lignes de vie (cf. Fig. 6) 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 (femme de ménage, petits-enfants). 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 !

*6.3.4 L'habitant auto-détourne ses usages ou comment un réfrigérateur devient un objet de communication interpersonnelle.* Les notes du journal de bord datées du début de l'installation du système font souvent référence à la quantification de notre habitat (*quantified home*). En particulier, nous avons appris le rythme de consommation énergétique de nos appareils ménagers, le réfrigérateur se révélant le plus gourmand. Dès lors, comment en diminuer la consommation ?

D'une part, nous utilisons notre propre création, les *fairy lights* de Fig. 1, comme afficheur périphérique de consommation énergétique (nous gagnons des lumières chaque fois que l'on éteint une lampe ou encore nous perdons des fleurs en fonction du budget consommé). D'autre part, nous mesurons 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 a suffi pour cela d'installer un capteur de contact sur la porte et de rédiger un programme qui repère les événements d'ouverture, envoie un email chaque fois que la porte reste ouverte plus de 20 secondes.

Nous avons d'abord pris conscience du nombre important d'ouvertures lorsque nous sommes deux à préparer le repas : le réfrigérateur se transforme en émetteur de *spams* ! Mais lorsque l'autre membre du foyer est encore au bureau, ces messages deviennent un moyen efficace de lui signifier qu'il est temps de rentrer : la porte est alors ouverte intentionnellement à d'autres fins que la préparation du repas ! Ces emails permettent aussi à celui qui est en mission au loin (décalages horaires) de se recaler sur la vie en France. Mais, ces avantages ont un prix : le coût attentionnel.

*6.3.5 Réduire le coût attentionnel est un défi.* Programmer, modifier les programmes, installer les équipements à façon, vérifier que le système fonctionne, demandent motivation et investissement cognitif qui viennent s'ajouter aux tâches usuelles de la vie courante. Il nous est arrivé 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.

## 7 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 [17], les expérimentations usuelles avec des participants représentatifs, ne sont pas non plus exemptes d'interprétations subjectives. *A minima*, on peut affirmer sans conteste que « *si ça ne marche pas pour soi, ça ne marchera pas pour autrui.*»

L'habitat domestique 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 aura conduit les études de terrain les plus avancées. Notre proposition, avec l'approche EUP/EUD, 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 [2]. 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, propriété phare en IHM, 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 [35]. Le thermostat Nest, incapable de reconnaître les exceptions, illustre au niveau d'un seul dispositif, les difficultés scientifique et technique à surmonter [53]. 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 sur son environnement [13, 6].

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 tout raisonnement et processus d'apprentissage puissent être 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 !

## REMERCIEMENTS

Ce travail a été soutenu par le projet européen 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, 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.

## REFERENCES

1. 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).
2. 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.
3. 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.
4. Brown, M., Coughlan, T., Ploetz, T., et al., (2015), "Editorial : Methods for Studying technology in the Home", Interacting With Computer, special issue on Methods for Studying Technology in the Home, 27(1), Oxford University Press, 1-2.
5. 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 Computer, 27(1), Oxford University Press, 60-71.
6. 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.
7. 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.
8. Carroll, J., Carrithers, C., (1984), "Training Wheels in a User Interface", Communication of the ACM, 27(8), 800-806.
9. 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.
10. 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.
11. Coutaz, J., and 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. Dey, A., Hamid, R., Beckman, C., Li, I., and 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.
19. 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.
20. 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, 311-326.
21. Fontaine, E., (2012), "Programmation d'espace intelligent par l'utilisateur final", thèse de doctorat, Uni. Grenoble Alpes, 204 pages.
22. Friedman, B., Kahn, P., and Borning, A., (1997), "Value Sensitive Design and Information Systems", Human Computer Interaction in Management Information Systems: Foundations, M.E. Sharpe, 1997.
23. Gaver, W., Dunne, A., and Pacenti, E., (1999), "Design: Cultural probes", Interactions, 6(1), ACM.
24. Green, T.R.G., M. Petre, M., (1996), "Usability Analysis of Visual Programming Environments: A Cognitive Dimensions Framework," J. Visual Languages and Programming, 7(2), 131–174.
25. 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.
26. IFTTT. https://ifttt.com.
27. 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.
28. 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.
29. Li, T., Li, Y., Chen, F., and Myers, B., (2017), "Programming IoT Devices by Demonstration Using Mobile Apps", In Proc. IS-EUD 2017, 6th International Symposium on End-User Development, Eindhoven, Springer.

30. Licklider, J.C.R., (1960), "Man-Computer Symbiosis", IRE Transactions on Human Factors in Electronics, (HFE-1, 4-11.
31. Lieberman, F., Paternò, F., Wulf, V., (2006), "End-User Development," Kluwer Academics.
32. Likert, R., (1932) A Technique for the Measurement of Attitudes, Archives of Psychology, vol. 140, 1–55.
33. 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.
34. 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 Computer, 27(1), Oxford University Press, 3-12.
35. Mozer, M.C., (2004), "Lessons from an Adaptive Home", Smart Environments: Technologies, Protocols, and Applications, John Wiley & Sons, Chapter 12.
36. Nardi, B., (1993), "A Small Matter of Programming; Perspectives an End User Computing", Cambridge MIT Press.
37. 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.
38. 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.
39. 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.
40. openHAB. www.openhab.org.
41. Poole, E., Comber, R., Hoonhout, J., (2015), "Disruption as a Research Method for Studying Technology in Homes", Interacting with Computer, Oxford University Press, 27(1), 13-20.
42. Resnik, M., Maloney, J., Monroy-Hernández, A., et al., (2009), "Scratch: Programming for all," Communication of the ACM, 52 (11), 60-67.
43. 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.
44. Rosen, N., Sattar, R., Linderman, R.W., Simha, R. and Narahari, B., (2004), "HomeOS: Context-Aware Home Connectivity," Proc. Int'l Conf. on Wireless Networks, CSREA Press, 739-744.
45. 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.
46. 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.
47. Terrier, L., Demeure, A., and 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.
48. 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.
49. 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.
50. 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.
51. 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.
52. Winograd, T., Flores, F., (1987), "Understanding Computers and Cognition : A New Foundation for design", Addison Wesley Publishing.
53. 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.