UsageGates
Propositions de travail autour de l'habitat intelligent
Le travail réalisé dans le groupe de Davidoff
Le travail autour de HomeOS (Microsoft)
End user Programming versus Agent Autonome
Objectifs de la réunion

Le travail réalisé dans le groupe de Davidoff

Scott Davidoff travail à CMU sur des sujets proches du notre, ces travaux sont intéressant à connaitre dans le cadre de notre travail. http://scottdavidoff.com/research Les recherches de Davidoff tournent autour des familles ayant plusieurs enfants. Son but est de proposer un habitat intelligent permettant de leur faciliter la vie . Ses recherches commencent un peu avant 2006 et se prolonge jusqu'à aujourd'hui sur le même thème. Il est intéressant de voir qu'il a passer les premières années à bien comprendre le contexte et à identifier les besoins des familles. De ces observations ont découlé des propositions de services qui sont du coup bien ancrés dans le contexte qu'il étudie.

Voici des commentaires sur une sélection de ses papiers :
Principles of Smart Home Control

http://dl.dropbox.com/u/10059910/scottdavidoff.com/publications/principles-of-smart-home-control.pdf

Ce papier marque le socle de ses recherches futures. Il y pointe que les habitats intelligents, de part leurs limitations actuelles (manque de perspicacité, rigidité, …) donnent un sentiments de pertes de contrôle à leurs habitants. En particulier, la rigidité des programmes qui sont déployés dans l'habitat entre en conflit avec la façon dont les gents vivent. Le comportement humain implique de large part d'improvisation et d'opportunisme (au bon sens des termes:)).

D'une manière plus générale encore, Davidoff constate que les familles ressentent déjà une perte de contrôle sur leur vie SANS qu'ils ne vivent dans des habitats intelligents. Cette perte de contrôle provient des nombreuses activités qu'ils doivent gérer (travail, école, famille, activités sportives et culturelles…).

A partir de ce constat, Davidoff analyse que le but d'un habitat intelligent (et de sa programmation) ne doit pas être de mieux contrôler les objets ou services mais de permettre aux habitants d'avoir un meilleurs contrôle sur leur vie. Pour cela, les systèmes de contrôles/programmations de l'habitat doivent être conçus de sorte à refléter la flexibilité observée dans les comportements humains.

La section 4.1 du papier liste les raisons pour lesquelles les familles ressentent une perte de contrôle sur leur vie. En résumé, cela tourne beaucoup autour de l'organisation des activités des enfants qui nécessitent beaucoup de flexibilité de la part des parents (ces activités ne sont pas des routines bien câblées), voir la section 4.2 du papier.

Davidoff propose une liste de sept principes que la programmation d'un habitat intelligent devrait respecter :

Permettre que les programmes (procédures, routines, …) puissent facilement (organiquement) évoluer. En gros, la vie d'une famille ne se réduit pas à quelques règles simples. Même partiellement, pour une activité à priori bien définit, il existe de nombreuses raisons pour changer de temps en temps la règle. Le système ne doit pas être rigide.

Pouvoir facilement spécifier de nouveaux comportements et encore plus facilement les modifier. Quotidiennement, les familles sont amenées à mettre en place de nouveau plan ou à modifier des plans existant pour tenir compte de nouvelles contraintes. Le système doit leur offrir un soutient dans la mise en place et le suivit de ces plans. On ne peut pas compter sur le fait que les personnes auront le loisir de programmer tranquillement ce genre de plans.

Comprendre les changements périodiques, les exceptions et l'improvisation. Les exceptions ne sont en fait pas exceptionnel, le système devrait prendre ça en compte. Comme les exceptions arriveront souvent mais ne seront pas prévisible, les routines doivent être très flexibles.

Prendre en compte dès le départ qu'il y aura de toute façon des problèmes. Dans la vie, il est rare que tout se passe comme planifié au départ, le système ne doit pas faire comme hypothèse que les plans seront bien suivit par les habitants.

Prendre en compte la multiplicité des buts, le fait qu'ils se superposent parfois partiellement et se contredise partiellement.

L'habitat est bien plus qu'un lieu. En gros, je pense qu'il veut dire que même à l'extérieur de l'habitat on devrait pouvoir accéder à des informations sur ce qui s'y passe, gérer les planning, gérer les listes de courses etc… Par exemple, la liste des courses à faire qui se trouve habituellement accrochée au réfrigérateur devrait aussi être accessible sur téléphone de l'extérieur.

La programmation de l'habitat participe à la construction de l'identité de la famille. Certaines activité liés à l'habitat sont partie intégrante de l'identité des personnes qui y vivent (ex : Bob s'occupe des plantes). Il faut prendre cela en compte dans les services proposés par le système de sorte que le système soit vu comme une aide et pas un « concurrent ».

Rapidly exploring application design through Speed Dating

http://dl.dropbox.com/u/10059910/scottdavidoff.com/publications/rapidly-exploring-application-design-through-speed-dating.pdf

Dans ce papier, Davidoff et son équipe présente Speed Dating, une technique de conception qui se place entre la phase d'idéation et la phase de prototypage. Il part du constat que les techniques UCD (User Centric Design) s'appliquent habituellement dans un contexte très différent de ce lui de l'UbiCom (Ubiquitous Computing) notamment sur 2 points: 1) il n'y a pas encore de vrai savoir faire disponible pour l'UbiComp qui permette de guider les concepteur et 2) le coût de développement d'un système UbiComp ou même d'un prototype semi fonctionnel d'un tel système reste très élevé. Ce dernier point pousse les équipe de développement à se focaliser très vite sur une solution au détriment des autres.

L'objectif de la méthode Speed Dating est de fournir un moyen rapide et bon marché pour comparer différents design en contexte. Cett méthode a été appliquer par l'équipe de Davidoff sur une centaine de design concepts et a permis de prototypé 27 variations d'applications au cours de 2 semaines. Une des différence de cette approche, mise en avant dans le papier, est que SD permet de structurer de nombreuses comparaisons entre des stratégies d'applications très variées (ou de nombreuses variations d'une même application dans un même contexte).

Le SD est divisé en 2 phases : Need validation et user enactments.

Need validation est dérivé de la méthode “concept validation” élaboré chez Philips. Cela consiste à placer les utilisateurs devant des storyboard papier qui illustrent les idées qu'ont eut les concepteurs après les études de terrains. Les utilisateurs commentent ces scénarios et cela permet, au travers de leurs remarques, de mieux comprendre sur quelles innovations les concepteurs doivent se concentrer. La phase de Need validation est là pour synchroniser les besoins qui ont été identifiés par les concepteurs avec ceux qui sont perçut par les usagers.

Pour chaque besoin recensé, on génère plusieurs concepts à même d'y répondre, ces concepts peuvent eux même faire surgir de nouveau besoins. Une fois qu'à chaque besoin correspondent plusieurs concepts.

A partir de là des scénarios sont conçut, dans le papier les auteurs indiquent que ces scénarios suivent le principe du “future breaching experiment” que je n'ai pas bien compris, il semble qu'il s'agisse de pousser les scénarios dans des cas un peu extrèmes mais pas trop non plus et ce afin de pouvoir mieux identifier les limites des concepts proposés… Les scénarios sont associé à une question “cadre” qui doit diriger l'entretien mené autour du scénario avec les utilisateurs. Cette question doit porter sur le besoin illustré dans le scénario.

Les discussions autour du scénario sont ensuite menées avec les utilisateurs. On focalise la conversation sur le besoin : qu'est ce qui le déclanche et quel est l'importance qu'il soit pris en compte à ce moment là.

On classe ensuite les besoins en termes de priorité. On synchronise les besoins recensé par l'équipe et ceux exprimés par les utilisateurs. Si des besoins exprimés diffèrent de ceux prévus, on reboucle pour mieux

les cerner. A la fin, on est censé avoir gagné une meilleurs compréhension des besoins et de la façon de les satisfaire. On comprend mieux au travers de la comparaison des multiples scénarios/besoins/concepts explorés.

User Enactment est la seconde phase au cours de laquelle les participants jouent leur propre rôle dans des scénarios établis à partir des données de la phase précédente.

Dans un premier temps, les concepteurs placent les problèmes de conceptions qu'ils veulent explorer dans une matrice. Les cases de la matrices sont peuplés avec des scénarios fictifs qui les illustrent.

Ces scénarios sont mis en places ensuite dans des lieux construits sur mesure avec des panneaux dessiné (ça ressemble à des sketches de maison grandeur nature…), les utilisateurs jouent leur rôle dans cet environnement, ce qui permet de mieux contextualiser les choses. Les utilisateurs ne sont plus dans une phase d'imagination de ce que donnerait le scénario mais ils le vivent. Ca se rapproche un peu de l'expérience design. Le SD est ici utilisé pour systématiser la mise en place de ces expériences…

Le travail autour de HomeOS (Microsoft)

Microsoft s'intéresse aussi à l'habitat intelligent. D'une part, des recherches sont menées autour de HomeOS, un système d'exploitation pour la maison. D'autre part, une de leur équipe travaille sur le projet Mayhem qui doit permettre à l'utilisateur de programmer ses appareils et différents services (multimédias mais pas seulement).
End user Programming versus Agent Autonome

An Adjustable-Autonomy Agent for Intelligent Environments

Un article intéressant permettant de fixer le débat sur l'opposition end user programming contre agent autonome : * End user programming uniquement : repose sur l'intelligence, la créativité et le bon vouloir ou l'implication de l'utilisateur qui peut ne pas avoir le temps par exemple. L'end user programming peux également aider l'utilisateur dans le cadre de handicap * Agent complétement autonome : l'agent et l'utilisateur travaille séparémment bien qu'ils doivent résoudre la même tâche. L'agent doit en plus deviner les intentions de l'utilisateur ce qui est difficile et peut conduire à un échec.

Il est préférable que les deux travaillent ensemble en équipe avec possibilité de régler avec un “curseur” le degré d'autonomie de l'environnement ou de controle du end user : Mixed-Initiative interaction * L'agent est composé d'un ensemble de règles associées à un degré de confiance. Une règle est d'autant plus active que son degré de confiance est élevé. Si le degré de confiance est trop bas, la règle disparaît. Si une règle est exécutée et a un impact (positif ?) sur l'environnement, sa confiance augmente. * La confiance d'une règle donnée par le système peut être fonction du nombre de fois que le système a vu l'utilisateur faire une action. * L'utilisateur et l'agent peuvent entrer tous les deux des

règles. En jouant sur la confiance initiale des règles de l'utilisateur d'un côté ou du système de l'autre, on règle le degré de collaboration entre les deux. * Si la confiance des règles entrées par le système est trop faible, il devra collaborer avec l'utilisateur pour augmenter cette confiance. Il devra en particulier demander à l'utilisateur s'il est d'accord avec la règle pour obtenir les points de confiance qui lui manquent. Il y a la possibilité aussi pour l'agent ou le end user de valider/modifier les règles de l'autre lors de leur ajout