Activity based End-User-Development for Smart Homes: Relevance and Challenges

Alexandre Demeure[a], Sybille Caffiau and Joëlle Coutaz[b]
[a] *INRIA, Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France*
[b] *Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France*

Abstract. Ubiquitous computing is now mature enough to unleash the potential of Smart Homes. The obstacle is no more about hardware concerns but lies in how inhabitants can build, configure and control their Smart Home. In this paper, we defend the idea that End-User-Development (EUD), which considers inhabitants as makers rather than mere consumers, is an effective approach for tackling this obstacle. However, to unleash its potential, we argue that considering the Smart Home as a big computer composed a sensors and actuators that can be weaved together is not enough. We propose to use Activity Theory as a structuring framework to guide the design of futures EUD systems. We reflect on the lifecycle of devices and services to discuss challenges that EUD system will have to address in the Smart Home context: installation and maintenance, designation, control, development (including programming and testing), and sharing.

Keywords. End User Programming, End User Development, Activity Theory, Smart Homes, Smart environments

Introduction

Ubiquitous computing has become a reality, even if not in the form that Weiser originally envisioned [17]. Widespread deployment of networks support interpersonal communication and enable people to access information such as news and encyclopedias, as well as services such as GPS-enabled navigation systems and weather forecast. This is made possible almost everywhere using smartphones, tablets or even laptops. Ubiquitous computing also takes place at home based on gateways such as ADSL modems and set-top boxes that provide Wi-Fi local networking with high-speed connection to the Internet as well as rich multimedia services including TV, audio and video sharing. Coupling these services with controllable wireless devices such as autonomous vacuum cleaners, makes it possible to build Smart Homes where quality of life is empowered by the dynamic interconnection of the physical world with the digital world.

The idea of making homes smart is not novel. Originally called "home automation", its development has been limited by the cost of the equipment, the lack of interoperability between proprietary communication protocols, and the difficulty to install and maintain the set up. Although the situation is improving, qualified technicians are still required for installing, modifying, and repairing home automation systems [2][10]. In addition, once installed, these systems provide a low added value

[1][15] and may show unexpected behavior that can lead to awkward situations for inhabitants. This state of fact prevents massive deployment or, at best, with limited application to specific domains such as energy saving and security [15].

As a consequence, the challenge of deploying ubiquitous computing applications in the context of the home is still opened. In this paper, we argue that combining End-User-Development (EUD) and Activity Theory [12][13] is a relevant approach to tackle this challenge. We then discuss the challenges to be addressed in order for this approach to be effective in the context of Smart Homes.

1. Relevance of EUD to Smart Homes

From a technical point of view, a Smart Home can basically be seen as a micro-cloud composed of several CPUs, data storage, sensors (e.g., luminosity, accelerometers and touch screens), as well as actuators such as lamps, autonomous vacuum cleaners, loud speakers, and screens. The combination of these components opens the way to a high and still untapped potential. Making the home smart is about deploying the right software (and hardware) in order to satisfy inhabitants' needs. We consider two non-mutually exclusive approaches to support this view: the "Smartphone reflex", and EUD.

The Smartphone Reflex. Whatever your need is, "there is an app for that". This approach, supported by tools such as Microsoft HomeOS [18], is attractive as it enables users to pick up what they need. In addition, users may incidentally retrieve useful apps they are not even looking for. Actually, this phenomenon is the basic reason for the use and success of app stores. However, this large world of apps can be difficult to manage in the domestic context when compared to the general use of smartphones.

First, the technical components of a smartphone are pretty well defined whereas those of Smart Homes are very diverse and unpredictable. Therefore, apps for Smart Homes have to address many sources of uncertainty, which is not an easy task. Second, there is one, possibly two, applications displayed at a time on a smartphone, whereas typical scenarios for Smart Homes envision a large number of services running in parallel with user interfaces distributed across the interaction resources available in the home. Finally, the smartphone is primarily for intimate use whereas, in general, the home is a shared space between several inhabitants.

The EUD approach. EUD is one approach to the "Do It Yourself" philosophy by providing inhabitants with the appropriate tools to build their home in accordance to their needs. Instead of relying on external developers who are supposed to elicit every possible fluctuant future need and, from there, to provide the "hypothetic right" software, EUD lets inhabitants form their own development team and from there, empowers them with enough room for personal creativity. In [11], Intille et al. observe that end-users are used and motivated to customize their personal devices. Thus, it is reasonable to hypothesize that people are willing to do the same for their home [3][6]. As suggested by Chin et al. [5], this customization could be all the more powerful and creative since appliances were deconstructed. The EUD approach is also well suited to address home heterogeneity: inhabitants are supposed to know what devices they own (although they might not be fully aware of what can be done with

them), and what components need to be programmed (at least partially). In addition, domestic constraints (e.g., who is allowed to access which services) may exist in the home. EUD is well suited to adapt home behavior to domestic constraints in a manner similar and consistent with other end-user programs.

As mentioned above, the two approaches, "the smartphone reflex" and EUD, are not exclusive: one beneficial combination of these approaches may be the use of apps as building blocks for EUD. The possibility to customize and combine devices, apps, and services is one of the key benefits from EUD, but it is also a key challenge [9]. Moreover, as pointed out by Davidoff [7], current EUD approaches fail to satisfy real user needs because they are too constrained and techno-centered. In the next section, we argue that EUD should be considered under the umbrella of Activity theory [13] to unleash its full potential in the context of Smart Homes.

2. Activity theory as a framework for positioning EUD in Smart Homes

Current EUD approaches merely consider Smart Homes as big machines composed of sensors and actuators that can be waved together to produce useful services. The effort is put on providing easy to use programming languages, may they be based on Event Condition Action rules [8], workflows or even magnets assemblies [19]. However, as noticed by [1][7][15], these approaches fall short at providing really useful support for home inhabitants as they do not take into account the inherent complexity, flexibility, improvisation and dynamicity of real life. We think that this is due to the underlying vision of the Smart Home as a big computer only [18], which is actually inherited from early home automation. We suggest that we have to change this underlying vision to make significant progress in satisfying people needs and think that Activity Theory can help us by providing an appropriate framework for positioning EUD in the Smart Home.

Activity theory is a conceptual framework originating from the socio-cultural tradition in Russian psychology. The foundational concept of the framework is "activity", which is understood as purposeful, transformative, and developing interaction between actors ("subjects") and the world ("objects"). The framework was originally developed by the Sovietic psychologist Aleksei Leontiev. A version of activity theory, based on Leontiev's framework, was proposed in the 1980s by the Finnish educational researcher Engeström. Since the early 1990s, Activity Theory has been used in the domain of Human Computer Interaction. In this article, we refer to activity theory as explained by Victor Kaptelinin and Bonnie Nardi in [12] and [13].

2.1. Hierarchical organization of activities

According to Leontiev, Human activities are units of life which are organized into three hierarchical layers (Figure 1). The top layer is the **activity** itself, which is oriented toward a motive, corresponding to a certain need. The motive is the object that the subject ultimately needs to attain. **Actions** are conscious processes directed at goals which must be undertaken to fulfil the object. Goals can be decomposed into sub-goals, sub-sub-goals, and so forth. Actions are implemented through lower-level units of activity, called operations. **Operations** are routine processes providing an

adjustment of an action to the ongoing situation. They are oriented toward the conditions under which the subject is trying to attain a goal. People are typically not aware of their operations.

This hierarchical structure provides us with some indications about how to employ EUD. **First**, activities are driven by motives that are proper to human subject (e.g. spend good time with friends). It is very unlikely that motive or activity can be directly "understood" by the EUD system but it definitely makes sense for inhabitant to have them explicitly represented in the EUD system. **Second**, actions and operations may be partially or totally (depending on the power of expression of the system) expresses with EUD (e.g. managing lights, playing music …). However, their raison d'être is intrinsically related to the activity in the sense that they support mediation between the subjects (inhabitants) and the object (the activity, the motive). The distinction between operation and actions (Figure 1) indicates that some programs could fully automate operations (e.g. turning lights on when a movement is detected) which is what is provided by current EUD systems, but that other programs would require interacting with humans to take into account improvisation or just because not everything can be automatized (e.g., choosing the music list to play). **Third**, relationships between actions and operations must not be seen as strict procedures that should be fulfilled by system and inhabitants. On the contrary, they are very flexible, letting place for improvisation and evolution [7]. EUD environments should enable inhabitants to quickly adapt actions, operations and their relationships.

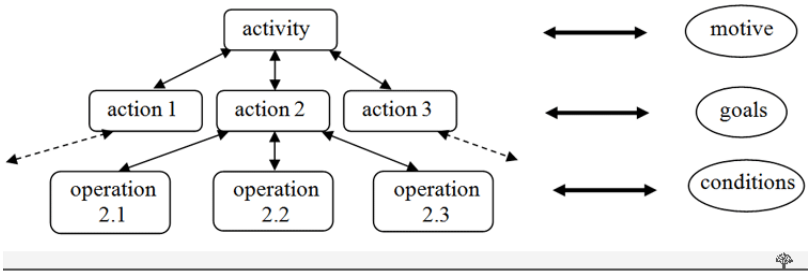

Figure 1. Hierarchical structure of activity, from [12].

2.2. Mediation is essential

Activity theory stresses the fact that there are mediations between subjects and objects: subjects use intellectual and/or physical tools (media) to achieve their activities. Activity theory inherits its special interest in mediation from the approach that made the most fundamental impact on Leontiev's framework: Vygotsky's cultural-historical psychology. Tool mediation allows for appropriating socially developed forms of acting in the world. Tools reflect previous experience of other people. Experience is accumulated in the structural properties of tools, such as their shape or material, as well as in the knowledge about how the tool should be used. Tools not only shape external behaviour, they also influence the mental functioning of individuals through internalization (e.g. automatic lighting change the way people move in the home and the mental model they have of it).

In the context of EUD for Smart Homes, mediation between the inhabitants and their motives should be supported by tools that enable direct control of devices and services, as well as tools for programming. What Activity Theory highlights here is that these tools should be grouped according to the activity they mediate. For instance, direct control of a living room media player and lights may be very relevant for the activity of "spending good time with friends" but not (or less) for "preparing kids to go to school". As a result, EUD should enable inhabitants to reference device, services and programs based on the activities they support. In other words, instead of providing a unique way to access devices and services, it should be possible to access them in multiple ways, depending on the activity. This could also mean that, depending on the activity, representations of these tools may vary. For instance, direct control of the media renderer may be more significant for the activity "watching a movie with family" than "spending good time with friends". Not only should EUD systems provide tools for inhabitant, they should also enable them to build their own tools (e.g. control panels and programs) and make them organically evolve with inhabitants' needs and experience.

2.3. Home inhabitants form a community

While Leontiev approach was primarily concerned with individuals, Engeström extended it by adding the notion of community (Figure 2). The **subject** – **object** relationship, mediated by instrument, is completed with the community. The relationship between **community** and **subject** is mediated by **rules** (e.g. within the family, there are rules of life) and the relationship between the **community** and the **object** is mediated by the **division of labor** (e.g. to keep the home clean, children have to tidy their room, parent have to clean the rest of the apartment).

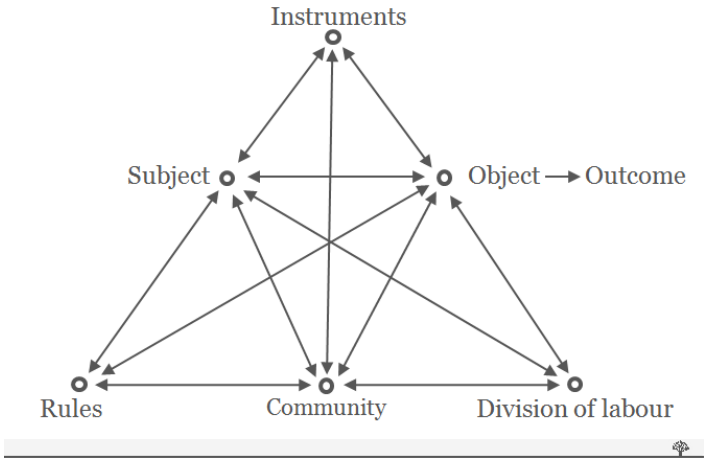

Figure 2. Engeström activity system model, from [12].

This extension to original Activity Theory is essential for us as homes are most of the time populated by several people (family, friends, etc.) trying to achieve common and individual motives that sometimes can be conflicting (e.g. "Keep house quiet so

that parent can rest" and "Lets' children have fun"). Making **Rules** (bottom left corner in Figure 2) explicit help us to consider that EUD may also provide tools support for inhabitant to implement these rules. For instance, if the activity is about "spending a nice evening in family", parents may have established the rule that children are not allowed to watch TV after 8PM (rule), or that cleaning the dinner table and washing up is an each turn process (division of labor). An EUD system could implement for instance the TV rule by enabling parents to program that after 8PM, it is no more possible to turn on TV using control panels (remote control, smartphone or whatever else) associated with children (rule). The "each turn process" division of labor could be supported by a service based on a calendar, registering who did what. What is really interesting here is the services and programs have a special purpose, different from others in the sense that they aims to support collectivity rules. Making their existence explicit may help inhabitants to handle the complexity of real life and provide more added-value.

In the same vein, division of labor helps us considering that some objectives in the home may be reach by combining the effort of several of its inhabitants. For instance, if the objective is to keep the home clean and pleasant, then this may be achieved by delegating floor cleaning to the father, washing-up to one child, maintenance of plants and the pets by the mother and on other child, etc. This EUD environment should take into account that activities can be shared, that programs may help in coordinating people to achieve on goal (e.g. it's your turn to wash the floor!).

2.4. Development of Smart Homes

As ordinary homes, smart homes evolve over time with renovation or acquisition of new everyday objects. In addition to this evolution, smart homes have a faster evolution due to the acquisition of new smart objects and services, so smart homes will be built incrementally both in terms of hardware and software. This is why development is a crucial aspect of smart homes.

Unfortunately, as pointed out by many researches [1][7][10][15], current home automation systems lack easy ways of evolution. The incremental development of Smart Homes results in several problems for the end-user: first, installing new devices or services then, later on, replacing or removing them. Once installed, inhabitants should be able to designate components in order to control them. Then, devices, apps, and services, can be interconnected by developing End-User Programs, which in turn includes programming, debugging, testing, maintaining, reusing, as well as sharing and retrieving from local data store or from public market places.

In addition to these challenges due to an incremental development, EUD has to respond to another one that activity theory brings out; smart home development is not a stable and linear process where services, devices and programs are one by one integrated and never questioned. Activity Theory states that activities and their subjects mutually determine one other. In other words, activities are generative forces that transform both subjects and objects. And then, stability cannot be reached once for all, each stability gives raise to new contradictions that in turn appeal for a new equilibrium (e.g. games and rules used by children help them to grow up but at some point enter in contradiction with the development of the child and need to be changed). An example of such contradiction coming from the HCI research field is the contradiction

between tasks and artefacts. The notion of "task-artefact cycle" [4] implies that the ultimate balance between tasks and artefacts cannot be achieved. A new artefact changes the task for which it is developed which means that another artefact needs to be developed to support the new task, and so on and so forth. That is also something that Davidoff notices when talking about the necessity to support organic evolution.

Activity theory requires that activities are always analyzed in the context of development. Development in activity theory is both an object of study and research strategy. As a research strategy, it prones to analyze the dynamic of how the object of study transforms itself over time in order to get a deep understanding of this object.

In the next section, we discuss challenges for developing an activity based EUD system in the context of Smart Homes as well as more technical challenges.

3. Challenges for an activity-based EUD in the Context of Smart Homes

The usual approach for EUD in smart homes consists on the one hand, in providing inhabitant with direct control over devices and services and, on the other hand, to enable them to program automation using a dedicated language and editor. Those points are essential, and should be contextualized with the Engeström Activity Theory framework to include more of the real life complexity and then, to provide more useful services. The first challenge is therefore how to efficiently handle activities in a EUD system.

3.1. Handling activities

In order to illustrate the challenges raised by activity handling in EUD system, we will take the example of a family composed with two parents and two children that would like to "spend good time with friends during a soirée". This example is deliberately chosen because it appears difficult to fully automatize such an activity and well illustrate the inadequacy of current EUD approaches. The EUD system cannot fully automatize this activity, collaboration has to be established with inhabitants and the system, some parts will be handled by one, the other or both. The underlying vision, quoted from Activity Theory, is that the Smart Home is a tool or set of tools. The use of this set of tools makes sense in the context of the activity (mediation) and is guided by inhabitants (subjects). In the rest of this section, we will roughly describe what end users would have to develop to support such an activity. The objective is not to describe a solution but rather to point out challenges that EUD system would have to face.

For this activity to be successful, the family prepares it in advance. However it is not likely that every details of the activity can be planned long-time in advance (a soirée is not a show), so the system should enable users to roughly define the activity and gradually giving it shape. For instance, in a first step, inhabitants could register the soirée on a calendar service and specify that the activity is composed of several goals (hierarchical decomposition of activity), namely "Prepare the apartment for the soirée", "control the ambiance" and "play games". Preparing the apartment for the soirée in turns can be decomposed in sub goals; "Clean the apartment" and "Prepare the meal" (define a menu, check for ingredient availability, reconstitute stocks, who

will cook, etc.). As the soirée activity has probably been done before, the system should enable the reuse and tailoring of existing activities. Before defining more in detail the activity, inhabitants may just want to define reminders to precise things further on. Hence, the first program will be a service to regularly remind that the soirée has to be prepared. The program could also trigger the reminders each time a parent goes to a supermarket.

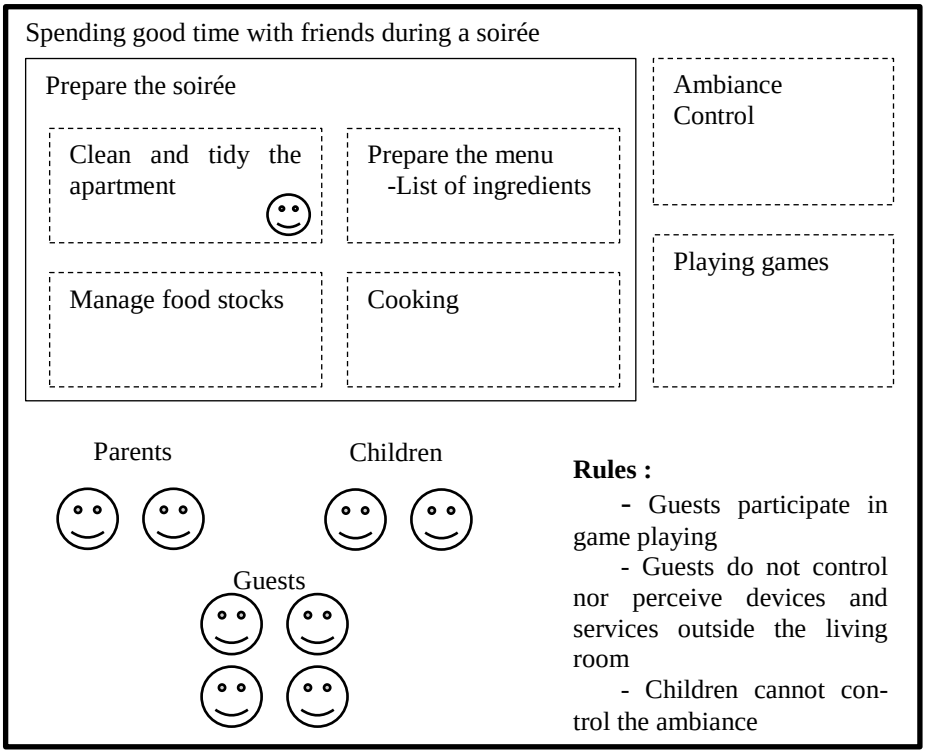

Figure 3. Sketch of an activity. Hierarchical structure of activity is represented by nested boxes. Goals are more or less precisely defined (plain or dashed lines). People participating to the activity are listed. Goals can be assigned. Focus is on the preparation of the soirée, the focus will evolve as soirée becomes closer.

Figure 3 is a sketch illustrating how an activity could be rendered. Not all the sub goals have to be immediately entirely specified (dashed boxes) as they can be precised later on. The nesting box structure mimics the hierarchical decomposition of activity, the more details you want about the activity, the more you have to zoom in. The focus is currently on the preparation of the soirée, this is the most urgent thing to do, and other parts of the activity (controlling ambiance and playing games) may be delayed later, so they appear smaller. People who participate in the activity are listed. Some rules and division of labor (in the sense of Activity Theory) can be defined for them. For instance, one parent is in charge of cleaning and tiding the apartment (division of labor), guests participate to games but have no control over the rest of the apartment (rule). Later on, the parent in charge of cleaning the apartment could make use of End User programs based on the calendar and autonomous vacuum cleaner to help achieving the goal (tool mediation).

Now let's suppose that we are getting closer to the soirée, the preparation is almost done, it's time to define more precisely what means "playing games". The family and the guest do really appreciate one particular game named "Time's up". In this turn-based game, two teams have to guess a maximum of personalities that are randomly selected in a time range of 60 seconds. The father would like to augment the physical game with an automatic minute glass playing a more and more stressing song till the end of time range. In order to achieve that, he can add a program to the "Playing games" goal: each time the father cellphone is checked, then play 'stressing music' on TV during 60 seconds. Another program could be specified so that scores can be displayed on the TV during the game. This last example illustrates that EUD should also enable inhabitant to quickly develop user interfaces and distribute them across rendering devices.

As a conclusion, this example illustrates that EUD for Smart Homes is not just about automating things once for all but should focus on providing tools for supporting activities. This support is going to be partial and evolving, the Smart Home should not be targeted to do things in place of its inhabitant but with them. In the next sections, we will discuss how turning this vision into reality also raises some more technical challenges.

3.2. Managing installation and maintenance

At first glance, installing a newly acquired component (i.e., a device, a service or an app), should be fully "plug and play", not to say fully "put and play" when it comes to wireless devices such as EnOcean enabled entities. In particular, no hardware configuration should be necessary to detect new components. Alas, there is more to consider.

Installing a component is a specific goal that aims at adding a new tool for inhabitant activities. Then, all performed tasks for installation prepare to other ones such as control and programming. In particular, there should be a mechanism to associate a "user-defined" name to components so that inhabitants can distinguish them more easily (for example, when the home includes several TVs from the same brand) or designate them later on as in "Switch off the TV of the living room". However, naming a component raises a number of questions: typically, in a multi-user home, should multiple names for the same component be authorized? Would it help each individual to better associate a meaningful name or would it provoke misunderstanding between the inhabitants?

In addition to support user-defined names, inhabitants should be able to associate any meta-data they think is useful for their daily life, such as the task for which the component has been acquired or, for photographs, the symbolic name of the place they have been taken (as opposed to a numeric meaningless geographic location).

The association of names and arbitrary meta-data to components implies that the system is able to identify the component technically and, from this technical identification (produced by vendors), generate the adequate User Interface (UI). As a possible solution, identification can be performed with a smartphone or a tablet augmented with technologies such as QR code, NFC or RFID tags.

Installing a component is not the end of the story. Inhabitants need to consider maintenance including replacing, renaming, moving or discarding a component. These modifications may have strong impacts on existing (and possibly running) end-

user programs and activities. If so, the Smart Home should be able to make these dependencies explicit and provide corrective actions.

3.3. Direct control

Once installed, components should be designated in order for users to control them either instantaneously in the real world (as we control TV sets using remote controllers), or asynchronously in end-user programs. Designation can be direct or indirect.

Direct designation relates to Norman's direct manipulation principles. It can be achieved in the same way as it has been performed for the install phase. It could be achieved by pointing at the component (if it is physical) or, if it is digital, by pointing at a representation of it (when it exists). Pointing can be done in many ways from using yet another device to free hand pointing gesture and eye gaze, the latter being far more challenging to implement reliably.

Indirect designation relates to Norman's language paradigm. It can be done using user-defined names combined with meta-data on which a selector mechanism can be applied. For instance, the availability of this mechanism would allow users to conveniently and concisely designate "all the lights of the living room", or "all the lights related to 'spending good time with friends' activity". This mechanism would be as useful and powerful for programming Smart Homes as CSS selectors are for designing webpages.

A "designatable" component can then be controlled. Here, the challenge is to make users aware of what can be done and how. In UI design, feedforward is the usual approach to support user's subsequent actions by representing the possible states the system can be in the future depending on the user's next actions [20]. Although a number of feedforward solutions exist for centralized UIs, there is no established solution for feedforward in the Smart Home where UI is, by essence, distributed. The other challenge appears when considering that, from the activity perspective, it is not one but many services, devices and even programs that have to be controlled because controlling on device alone rarely make sense (e.g. "spending good time with friends" would typically imply controlling all lights of the living room plus the media renderers and programs dedicated to set ambiances).

3.4. Languages and tools for programs development

Development is at the core of EUD. It implies designing adapted programming languages integrated in a well-thought out programming environment that should at least support testing and debugging. [8] have shown that users are quite at ease with the Event Condition Action (ECA) rule-based paradigm. However, ECA programming is known to be challenging when the number of rules is large. In order to reduce the ECA space, should we propose means for grouping rules? The hierarchical structure of activities, as described above, could help grouping ECA rules in a relevant way.

There is also a need for high level abstractions: the most obvious lesson from surveys is the tendency for interviewees to be goal and behavior-centric, but not techno-centric [19]. The language should therefore support this form of reasoning and let users express routines instead of procedures [7].

In addition, the vocabulary of the language is not going to be established once for all. Part of the vocabulary will be provided through devices and services descriptions (e.g. for a lamp: "turn it on/off") as new devices and services are integrated. But there is more than that, the EUD system should also be able to recognize a new situation based on what it can sense in the home. Indeed, it is very improbable that inhabitants will be able to associate combinations of sensory measures with situations due to the inherent complexity of such a task. As a consequence, EUD system should be able to learn from what it observes and propose inferred situations to inhabitants so that they have the opportunity to correct, retain, discard, and name them. This raises an important challenge in terms of visualization: how to render a new system-discovered situation (which may be dynamic) to users?

As for any language, there is a tradeoff to make between expressiveness and simplicity. For instance, should the language offer mechanisms for abstracting rules or group of rules? Should it be possible to define variables and parameters? Time is an important aspect for programs, how should it be integrated in the language? Should we rely on metaphors like calendar [16] or explore more abstract representations?

We think that the language should at least be expressive enough to be able to define domestic policies. This would imply to be able to specify access rights to services and devices, but also to define policy for conflict resolution (e.g., if two rules are contradictory, then "select those from the parents, and if this not possible, ask the parents what to do"). In other words, we believe that policy should not be hardcoded in the system, but specified by end-users. For the sake of consistency, policy should be expressed in a similar way than any other program.

Testing and debugging. A very important way to support the inhabitants is to offer a programming environment which enable to visualize, test and debug programs. It is fundamental to explain the links between the programs and the actions: what are the rules that induced a specific behavior of the system [14]. Thanks to this kind of quick test and simulation tools, design and programming can be iterative and consequently best adapt to users. Besides, Holloway et al. [10] confirm this user desire, among others, of iterative development. They also affirm that it seems very important for potential users to have a system which enables iterative usages. For example, after the system installation or some policies' development, an adaptation and adjustment phase is usually observed [15] and can lead to device repositioning or policies rewriting.

3.5. Sharing experience and social programming

One last challenge is to support emulation inside and between homes. Between homes, the emulation could be supported via forums (where Smart Homes inhabitants already spend time to share their experiences), open market places for sharing and exchanging components, or social programming where users express needs and ask questions while others provide hints, tips and tricks.

Inside the home, an effort should be made to give its place to non-programmers. By non-programmers, we mean inhabitant that are not the local high-tech guru who actually use the system. Too often they feel trapped in their own home because of unexpected behavior and most of all the feeling that they cannot change things. At least the EUD system should enable them to express their requests

4. Conclusion

Current technologies are mature for ubiquitous systems but ubiquitous systems in homes remain latent. Actually, the problem is to enable the users to express their needs and to give them the relevant tools to solve them using technology in their homes. We think that an End-User Development based on the principles of the Activity theory brings foundations for a solution that increase the cohabitation between inhabitants and programs in smart homes although several challenges that we identified have to be overcome.

Acknowledgements. This work has been supported by the European Catrene project AppsGate and AmiQual4Home, ANR-11-EQPX-0002.

References

[1] Bernheim Brush A.J. et al. (2011). Home automation in the wild: challenges and opportunities. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11).
[2] Björkskog, Christoffer. "Human Computer Interaction in Smart Homes.".
[3] Blase Ur. et al. Practical trigger-action programming in the smart home. CHI 2014.
[4] Carroll, J. M. (ed.) (1991): Designing Interaction: Psychology at the Human-Computer Interface. Cambridge, UK, Cambridge University Press.
[5] Chin J, Callaghan V, Clarke G, "Soft-appliances: A vision for user created networked appliances in digital homes", Journal of Ambient Intelligence and Smart Environments 1 (2009) pp 65–71.
[6] Coutaz, J. et al. (2010) DisQo : A user needs analysis method for smart home in Proceed-ings of ACM NordiCHI 2010 International Conference, pages 615-618.
[7] Davidoff, S. et al. (2006). Principles of smart home control. Lecture Notes in Computer Science, 2006: Ubiquitous Computing (Ubicomp 2006), 4206: 19-34.
[8] Garcia-Herranz, et al. Towards a Ubiquitous End-User Programming System for Smart Spaces. Journal of Universal Computer Science, 16(12), 012, 1633-1649.
[9] Holloway, S., & Julien, C. (2010, November). The case for end-user programming of ubiquitous computing environments. In Proceedings of the FSE/SDP workshop (pp. 167-172). ACM.
[10] Holloway, S., Stovall, D., & Julien, C. (2009). What Users Want from Smart Environments. Technical Report, UT-EDGE-2009-008.
[11] Intille, S. (2002). Designing a home of the future. Pervasive Computing, IEEE, pp76-82.
[12] Kaptelinin, V. (2013): Activity Theory. In: Soegaard, Mads and Dam, Rikke Friis (eds.). "The Encyclopedia of Human-Computer Interaction, 2nd Ed.". Aarhus, Denmark: The Interaction Design Foundation. Available online at http://www.interaction-design.org/encyclopedia/activity_theory.html
[13] Kaptelinin, V. and Nardi, B. A. (2012): Activity Theory in HCI: Fundamentals and Reflections. Morgan and Claypool.
[14] Ko, A. J. and Myers, B. A. (2008), "Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior", pp76-82.
[15] Mennicken, S. and Huang, E. M., « Hacking the Natural Habitat : an in-the-wild study of smart homes, their development, and the people who live in them » (2012), Pervasive'12 Proceedings of the 10th international conference on Pervasive Computing, pp143-160.
[16] Mennicken, S. et al. (2014) , Casalendar: A Temporal Interface for Automated Homes, CHI 2014.
[17] Rogers, Y. 2006. Moving on from weiser's vision of calm computing: engaging ubicomp experiences. In Proceedings of the 8th international conference on Ubiquitous Computing (UbiComp'06), Paul Dourish and Adrian Friday (Eds.). Springer-Verlag, Berlin, Heidel-berg, 404-421.
[18] Rosen, N., Sattar, R., Linderman, R. W., Simha, R., & Narahari, B. (2004, June). HomeOS: Context-Aware Home Connectivity. In International Conference on Pervasive Computing and Applications.
[19] Truong, K., Huang, E., & Abowd, G. (2004). CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. UbiComp 2004: Ubiquitous Com-puting, 143-160.
[20] 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.