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

Rémy Dautriche, Camille Lenoir, Alexandre Demeure, Cédric Gérard, Joëlle Coutaz, Patrick Reignier

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

R. Dautriche[1], C. Lenoir[1], A. Demeure[1], C. Gérard[1], J. Coutaz[2] and P. Reignier[1]

[1] PRIMA, INRIA, LIG, Universités de Grenoble
{remy.dautriche,camille.lenoir,
alexandre.demeure,patrick.reignier.inria.fr}@inria.fr
[2] IIHM, LIG, Universités de Grenoble
joelle.coutaz@imag.fr

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. 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, testing, and reusing), and sharing.

Keywords: End User Programming, Smart Homes, Smart environments.

1 Introduction

Ubiquitous computing has become a reality, even if not in the form that Weiser originally envisioned [10]. 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 sys-

tems [1][6]. In addition, once installed, these systems may show unexpected behavior. This can lead to awkward situations for inhabitants, preventing massive deployment or, at best, limiting application to specific domains such as energy saving and security [9].

As a consequence, the challenge of deploying ubiquitous computing applications is still opened. In the rest of this paper, we argue that End-User-Development (EUD), by providing the right tools that turn inhabitants into makers instead of consumers, is a relevant approach to tackle this challenge. We then discuss the challenges to be addressed in order for EUD to be effective in the context of Smart Homes.

2 Relevance of EUD to Smart Homes

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 [11], 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 let inhabitants form their own development team and from there, empowers them with enough room for personal creativity. In [7], 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 [2]. This

approach is also well suited to address homes heterogeneity: inhabitants are supposed to know what devices they own, and what components need to be programmed. In addition, EUD is well suited to define domestic policies (for example, how to solve a conflict between programs, and who can access which services) 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 [5]. In the next section, we discuss this challenge as well as other challenges that appear when considering the lifecycle of Smart Home components.

3 Challenges for EUD in the Context of Smart Homes

As ordinary homes evolve over time with renovation or the acquisition of new everyday objects, so Smart Homes will be built incrementally, both in terms of hardware and software. 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** these 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.

3.1 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[1] enabled entities. In particular, no hardware configuration should be necessary to detect new components. Alas, there is more to consider. 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).

[1] http://www.enocean.com/en/home/

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. If so, the Smart Home should be able to make these dependencies explicit and provide end-users with corrective actions.

3.2 Designation

Once installed, components must 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". This mechanism would be as useful and powerful for programming Smart Homes as CSS selectors are for designing webpages.

3.3 Control

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 [13]. 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.

3.4 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. Garcia-Herranz et al. [4] have shown that users are quite at ease with the rule-based paradigm. However, rule-based programming is

known to be challenging when the number of rules is large. In order to reduce the rules space, should we propose means for grouping rules?

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 [12]. The language should therefore support this form of reasoning and let users express routines instead of procedures [3].

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") but new words may appear to name situations that were not predicted at design time (e.g., "dinner with friends") but that make sense not only for inhabitants but for the system as well. By this, we mean that the system should be able to recognize a new situation based on what it can sense in the home. 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, a 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? We think that the language should at least be expressive enough to be able to define domestic policy. 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 for any other program.

Testing and debugging. Testing and debugging is unavoidable in any software development activity. In the context of EUD for home, it is is fundamental to show the links between the program elements and the system behavior (e.g., which rules have resulted in such and such system behavior? [8]), so that users can answer "why" questions. Users also need to answer "what if" questions. In turn, these questions call for a dual digital representation of the home so that users can test their program for situations that would be difficult (not to say unsafe) to test in the real world (such as "if there is smoke in the kitchen and there is no one at home, call everyone's smartphones"). The development of such a simulator is in turn a challenging issue.

3.5 Sharing experience and social programming

One last challenge is to support emulation between homes. This could take the form of forum, open market places for sharing and exchanging components, or social pro-

gramming where users express needs and ask questions while others provide hints, tips and tricks.

4 Conclusion

The enabling technologies for ubiquitous computing are reaching maturity. However, the application of ubiquitous computing for the home remains latent. The problem is to empower users with tools that allow them to configure and control their home. We believe that EUD can provide the foundation for such tools. However, as demonstrated in this paper, a number of key challenges must be overcome.

Acknowledgements. This work has been supported by the European Catrene project AppsGate.

5 References

1. Björkskog, Christoffer. "Human Computer Interaction in Smart Homes."
2. Coutaz, J. et al. (2010) DisQo : A user needs analysis method for smart home in Proceedings of ACM NordiCHI 2010 International Conference, pages 615-618.
3. Davidoff, S. et al. (2006). Principles of smart home control. Lecture Notes in Computer Science, 2006: Ubiquitous Computing (Ubicomp 2006), 4206: 19-34.
4. Garcia-Herranz, et al. Towards a Ubiquitous End-User Programming System for Smart Spaces. Journal of Universal Computer Science, 16(12), 012, 1633-1649.
5. Holloway, S., & Julien, C. (2010, November). The case for end-user programming of ubiquitous computing environments. In Proceedings of the FSE/SDP workshop on Future of software engineering research (pp. 167-172). ACM.
6. Holloway, S., Stovall, D., & Julien, C. (2009). What Users Want from Smart Environments. Technical Report, UT-EDGE-2009-008.
7. Intille, S. (2002). Designing a home of the future. Pervasive Computing, IEEE, pp76-82.
8. Ko, A. J. and Myers, B. A. (2008), "Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior", pp76-82.
9. 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.
10. 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, Heidelberg, 404-421.
11. 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.
12. 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 Computing, 143-160.
13. 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.