The Case for End-User Programming of Ubiquitous Computing Environments

Seth Holloway and Christine Julien

The Mobile and Pervasive Computing Group,
The Center for Excellence in Distributed Global Environments,
The University of Texas at Austin

TR-UTEDGE-2010-015

The Case for End-User Programming of Ubiquitous Computing Environments

Seth Holloway and Christine Julien
Mobile and Pervasive Computing Group
The University of Texas at Austin
Austin, TX, USA
{sethh, c.julien}@mail.utexas.edu

ABSTRACT

Gone are the days that computers will be used by select users sitting at a desk with a mouse and keyboard. The next wave of computing, ubiquitous computing, is upon us. With smart phones, tablet computers, and embedded sensors/actuators flourishing, users are already interacting with dozens of computers per day. A large body of research has addressed many issues in hardware and software for the future, but few have focused on the users. We posit that the reason ubiquitous computing environments are still largely unrealized is because research is technology-centric, with inadequate focus on users. To bridge this gap between what technology can provide and what users need and want from ubiquitous computing, we motivate the need for end-user programming in ubiquitous computing environments and provide a vision for enabling end-user programming. We believe that the software engineering community must provide end-user programming capabilities in ubiquitous computing environments if this domain is to reach its full potential.

1. INTRODUCTION

Ubiquitous computing has been a heavily researched area since the late 1980s [19], yet the predicted "third wave of computing," a world full of computers that calmly better our lives [18], has fallen short. Indeed, ubiquitous computing environments stand to revolutionize the way we live. With the envisioned smart homes, smart workplaces, smart construction sites, etc., no human space will be left "dumb." Unfortunately, our spaces are in fact still mostly dumb. Truly, after great academic research and industrial efforts, ubiquitous computing has become a reality—just not in the way that was initially predicted. Computers are everywhere! In cellular phones, cars, washers and dryers, refrigerators, TVs, and many more everyday devices. Missing is the software to federate these devices and make them work for us.

Perhaps the most commonly cited example of a future ubiquitous computing environment is the smart home, where sensors and actuators are embedded throughout the residence to monitor a diverse set of properties. Readings may be coordinated to infer high-level conditions about the state of the home and its occupants. Based on these conclusions and some set of configured behaviors, a smart home can affect changes to make the home's occupants more comfortable, more productive, or to reduce costs. Other smart environments follow similar high level flows (where sensed conditions trigger actions), but their specific implementations and goals will differ. One of the primary challenges in building these smart spaces is the massive amounts of customization necessary; the process of actualizing user needs and expectations in hardware and software is not trivial. Due to the large and varying individual needs for these applications, the traditional software paradigm will not scale. To create individualized environments, users cannot simply buy a COTS solution and expect it to immediately satisfy their needs.

We must provide extensible architectures that allow users to program their own smart environments. The architecture should build on well understood web protocols, and research should actively and continuously engage users. To support this position, we first investigate some initial forays into supporting development of ubiquitous computing applications; then, we look at how end-user programming has been successfully applied in other domains. We then move on to some initial directions and our vision for end-user programming applied to ubiquitous computing. Specifically, we report on a formal survey of end users to start to uncover how they "think" about intelligent environments so that an end-user programming approach can be rationally structured.

2. UBIQUITOUS COMPUTING'S NEED FOR END-USER PROGRAMMING

With large amounts of money and expertise, some form of intelligence could be added to any space. Today's commercial systems are extremely human-intensive, requiring large amounts of custom software development and configuration. Commercial systems are controlled by the vendor using proprietary, rather than open, standards. This prevents end-users from extending their own environment; instead, users have to go through the seller for any changes. This approach is fundamentally flawed as it is unlikely that any single company could provide every sensor/actuator necessary for the diverse ubiquitous computing applications worldwide.

There are many academic approaches that aim to make ubiquitous computing a reality. We can place these approaches on a spectrum based on the amount of human involvement. To one extreme is machine learning, which places decision-making entirely in the hands of the system;

FSE '10 Santa Fe, New Mexico USA

at the other extreme is end-user programming, which asks the user to personally "program" the environment.

Machine learning can eliminate the need for humans in the loop by automatically learning an inhabitant's routine. CASAS [13] provides a framework to automatically learn behaviors but also provides the user with basic feedback abilities. The user involvement is wonderful, but there are additional needs to accommodate the full needed range of expressiveness. Early activity recognition schemes [16] were imprecise, with prediction accuracies as low as 25%. Other approaches [2] have improved accuracy to at least 47% and now achieve roughly 80% accuracy—a fine figure, but still lacking with respect to satisfying end-users. These machine learning approaches also require preset information that may be too detailed for users to provide and limits the open-endedness and extensibility ubiquitous computing environments require. We believe that these machine learning approaches must be deployed in conjunction with end-user programming that can bolster the correctness and usability of the system. For example, if the system senses events not interesting to users, users could quickly and easily filter out and disable those capabilities. Machine learning is also difficult, if not impossible, when events are very rare (for example, a burglary) or very complex (for example, home healthcare where mistakes could have catastrophic results).

A number of projects have explicitly aimed to involve the end-user in customizing the environment. Media Cubes [1] offers a tangible programming interface similar to remote controls. Inhabitants associate each face of a cube with a specific device's action, for example, turning on the television. Users can later perform the associated action by turning the cube to the desired face. In CRISTAL [3] users interact with a tabletop displaying a live video feed of the space. The system is programmed to provide cues for interacting with each device, such as a play icon superimposed on the television. This allows users to easily understand what they are controlling, but there is no guarantee that the superimposed control will be natural or even any notion of what is "natural" in this space. Extensibility is also an issue; for example, what symbol will be superimposed on a new actuator placed in the environment? In "Playing with the Bits" [6], users snap available components together using a jig-saw puzzle metaphor. Unfortunately, the subset of devices (represented as puzzle pieces) is limited by what the researchers can produce. CAMP [17], created a "magnetic poetry" interface that mimics refrigerator magnets. These magnets offer a set of words that participants can rearrange into phrases. This provides a different interface for automated capture and playback (which allows users to replay events that were automatically recorded in the home).

Whether the projects involve the users a lot or a little, previous research into software systems for ubiquitous computing has typically targeted the enabling technologies required to sense and act upon the environment. This *bottom-up* approach has enabled the development of powerful applications. However, the motivation for these applications is traditionally provided by the developers and researchers as a means to demonstrate their new engineering or technology feats. There is not commonly a formal connection between the *programming* task and its abstractions and the *users* of the environments. We feel the mismatch between these applications and the expectations of mainstream users slows adoption. It is our position that the community requires a *top-down* approach to ubiquitous computing—one that is centered around users. This is inherently an *end-user programming* approach; in the next section, we will explain how end-user programming has been adapted in other domains to create powerful and successful platforms.

3. EXISTING END-USER PROGRAMMING

It is estimated that by 2012 there will be 90 million end-user programmers [15]. This number includes traditional programmers (roughly 13 million) as well as workers who use spreadsheets and databases; however, people are now performing similar programming feats as they extend functionality in their favorite applications and platforms, making virtually everyone an end-user programmer. In this section, we detail some systems that allow for such end-user customization and programming. By applying lessons learned from these systems, we believe we can quickly create usable ubiquitous computing environments.

Mobile Devices, Browsers, and Productivity Tools. The widespread use of smartphones has made it evident that end-users are well-motivated to add functionality to their devices by updating and adding new applications. Browsers have become widely extensible with *add-ons*, and productivity tools include macros that enable users to add behavior to documents.

Databases and Spreadsheets. Traditionally, end-user programming has focused largely on non-programmers' interactions with databases and spreadsheets. For example, a SQL statement like `SELECT * FROM Users` can be created by someone with rudimentary computer skills thanks to straightforward interfaces that abstract low-level database complexities and unfamiliar languages. Similarly, spreadsheet interfaces allow virtually anyone to create advanced mathematical formulas and perform analyses without implementing the calculation functions. Similar end-user interactions can be seen in several other prominent systems where people extend functionality without using traditional programming methods.

Games. Video games have also had great success with allowing users to customize the application with extensible platforms and interfaces that transform game internals into easily edited graphical elements. For example, *World of Warcraft*, a hugely popular[1] roleplaying game, allows users to customize the user interface and extend functionality by creating bundles programmed in Lua. To date, over 6,700 such extensions have been created and shared by users.

Diverse applications in domains from entertainment to productivity have embraced end-user programming and customization. End-users have demonstrated their comfort with this paradigm; we believe this demonstrates that they are ready and able to program their own ubiquitous computing environments. In the next section we discuss our vision for a potential way forward with end-user programming applied to the envisioned ubiquitous computing environments.

4. THE WAY FORWARD

To start to get a handle on how an end-user programming framework for ubiquitous computing would best be structured, one must first ascertain how end-users think and

[1]In 2008 there were over 11.5 million subscribers worldwide (`http://us.blizzard.com/en-us/company/press/pressreleases.html?081121`)

talk about their intelligent environments—that is, what they want from ubiquitous computing.

4.1 Determining What Users Want From Smart Environments

Many others (including [8, 10, 11, 12, 14]) have surveyed what users need and want from smart environments. With a goal of capturing the users' mental model of smart home actions, we performed a large-scale user study [4]. We chose smart homes because they are an easily understood instantiation of ubiquitous computing. We asked users to write three different policies for their smart home to follow: two structured policies and one open-ended.

In 10 days online we received 64 completed responses (62 from the continental United States). Overall, people were very excited about smart homes and the potential impact of smart environments, such as this user: *"[Smart homes] sound very groovy and could make life a lot easier."* And another user said, *"Now someone just needs to make it easier to create smart homes."* In fact, many people begged for simplicity and demanded a simple user-centered programming interface for smart homes. We also found that despite a great deal of interest among academics, very few users, only 6%, expressed concerns for their privacy. One user stated that the idea of the home having knowledge of his schedule was *"Too Big Brother for me."*

To formalize end-users' mental model of smart environments, we wondered *Are there any specific words or sentence constructs used by respondents?* To answer this question, we analyzed the responses and found that conditional logic was an extremely popular method of relaying smart home behaviors. A majority of respondents (73%) used some form of conditional logic ("if," "then," "else," "while," "when") to describe the scenarios. This way of reasoning was expected from highly technical people (for example, programmers who use similar reserved words), but the use of conditional logic was surprisingly common across all demographics.

While we gained many valuable insights from the survey, two things were abundantly clear after reviewing responses:

- Different users desire vastly different behaviors from their space, and
- Different users conceive of performing similar actions in very different ways.

Very few users (5%) detailed situations described in research; however, many wanted different vastly behaviors. For example some users thought of their pets,

> *A system that senses that [the dogs] have emptied the water dish (inside and out) and refills the dish. Sometimes they drink lots and sometimes during the day one of the dogs likes to play in the water so they end up with very little water...*

while others thought of automated cleaning:

> *Make bathrooms "self-cleaning" like those cool port-o-potties in Paris. They close up like a dishwasher and steam everything...*

We expected many users would repeat our example scenarios, but few people did so. Therefore, we believe that it is not hardware capabilities that are holding back ubiquitous computing but serious software integration and usability issues. Coupled with the fact that the end-users specified highly personalized scenarios, this demands straightforward, abstract, and usable end-user programming constructs tailored for ubiquitous computing. People were very interested in automating their lighting, heating, and cooling. To make the home comfortable, some people imagined opening windows or closing blinds at the right time; others chose standard air conditioning; others used even more complex multi-device interactions. The most common applications that users described required functionality that is already commercially available, albeit obscure and expensive.

4.2 Vision

Our vision for the future of ubiquitous computing software follows these three principles:

- Involve users in the design process,
- Create one extensible platform that accomplishes the high-level task of mapping conditions to actions, and
- Build on powerful, widely deployed web protocols.

To speed adoption we must engage users in the design process or the system will never be, well, usable. To be maximally effective, the platform should be human-centric and only moderately abstract [9]. The traditional software conception of "programs are logic" is too computer-centric and thus unusable by most users. If the platform is is too concrete, it cannot not satisfy the wide range of possible devices; if the implementation is too abstract it will negate users' ability to reason about the program. As a first step, we should embrace the lessons learned by the personal computing systems mentioned in Section 3 and informed by the language mined from our structured survey. Not only is the extensibility technologically sound, it also provides an important sociological effect: the ability to customize engages users and improves the user experience [7].

Ubiquitous computing has visions covering virtually all facets of life, ranging from the workplace to the home to construction sites and beyond. At the core, these ubiquitous computing environments simply perform actions when conditions are met. The specific actions and conditions are limited by the hardware available; for example, without a light sensor, the environment cannot deterministically perform actions based on light level. Similarly, a smart environment cannot sound an alarm without an alarm present. Smart construction sites will have different hardware than smart homes, just as the devices in an elderly person's smart home will differ from that of a healthy college student. Unfortunately, the scope in ubiquitous computing is nearly infinite: an ever-expanding catalog of parts can be combined in any combination. The mapping of conditions to actions will be unique to each smart environment, so completely general purpose solutions are infeasible. While *domain-specific* frameworks have been studied (namely, smart homes), even within a domain, the hardware and software combinations are so large to make these solutions inflexible. Instead, we envision one extensible software framework that allows additional software and hardware functionality to be plugged in—mirroring the systems described by the survey participants. Rather than creating and maintaining two completely separate software solutions for two different smart environments, we should use the same general platform with different sets of extensions. As in current systems, these extensions could be created by users and made freely available.

To further ease development, building this ubiquitous computing platform on web protocols enables tremendous benefits. The Internet is arguably the most important tech-

nology of our time, and it will only continue to gain prominence. Web protocols already have wide deployment, robustness, supporting frameworks, millions of skilled developers, terabytes of tutorials, free updates, and phenomenal flexibility thanks to the widespread interoperability.

It is not necessary for ubiquitous computing solutions to be on the Internet (in fact, many people may explicitly *not* want their smart spaces to be accessible from anywhere), but we believe that the solutions should build on web technologies. A web-enabled platform is easily achievable as devices simply need to communicate using HTTP. We have shown that using such web technologies to support ubiquitous computing application development can be made straightforward and approachable [5]. Devices, including household appliances like washers and dryers, are now shipping with this capability, so we just need the software to bring these appliances under administrative control. The platform server could be written in any of the myriad web languages (Perl, PHP, Java, Python, Ruby, ASP.NET, etc.) or their web application frameworks. This diversity of languages provides incredible strength and extensibility that will drive the growth of ubiquitous computing.

5. CONCLUSIONS

Ubiquitous computing is the present and the future. Personal computing will persist, but we must focus more attention on ubiquitous computing. Our vision for the future of ubiquitous computing software follows three principles: (i) *involve users in the design process*, (ii) *create one extensible platform that accomplishes the high-level task of mapping conditions to actions*, and (iii) *build on powerful, widely deployed web protocols*. We believe that following these principles will provide the most robust, shortest path forward. Smart environments will not flourish unless end-users are able to tailor systems to their liking; in the domain of ubiquitous computing, this means mapping conditions to actions, requiring expressive programming constructs. The technology necessary to create smart environments (sensors and actuators, computers, wireless networking, etc.) is available—or will be—in the near-future; however, without further research into users' desires and capabilities and enablements for user programming of ubiquitous computing environments, smart environments will not be widely adopted. By focusing on users, smart environments will move from an academic vision to a hugely useful reality.

Acknowledgments

The authors would like to thank EDGE for research facilities. This work was funded, in part, by the National Science Foundation (NSF), Grant # CNS-0620245. The views and conclusions herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

6. REFERENCES

[1] A. Blackwell and R. Hague. AutoHAN: An architecture for programming the home. In *Proc. of the IEEE Symp. on Human-Centric Computing Languages and Environments*, pages 150–157, 2001.

[2] D. Cook, M. Youngblood, E. Heierman III, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. MavHome: An agent-based smart home. In *Proc. of the 1st Int'l. Conf. on Pervasive Computing and Communications*, pages 521–524, 2003.

[3] M. Haller, P. Brandl, C. Richter, T. Seifried, J. Leitner, and A. Gokcezade. Interactive displays and next-generation interfaces. In *Hagenberg Res.* 2009.

[4] S. Holloway, D. Stovall, and C. Julien. What Users Want from Smart Environments. Technical Report, UT-EDGE-2009-008, 2009.

[5] S. Holloway, D. Stovall, J. Lara-Garduno, and C. Julien. Opening pervasive computing to the masses using the seap middleware. In *Proc. of the Middleware Support for Pervasive Computing Workshop*, 2009.

[6] J. Humble, A. Crabtree, T. Hemmings, K. Åkesson, B. Koleva, T. Rodden, and P. Hansson. "Playing with the Bits" User-configuration of Ubiquitous Domestic Environments. In *Proc. of Ubicomp*, 2003.

[7] A. J. Kim. *Community Building on the Web: Secret Strategies for Successful Online Communities*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[8] S. Kim, S. Kim, and H. Park. Usability challenges in ubicomp environment. *Proc. of Int'l. Ergonomics Assoc.*, 2003.

[9] A. Ko, B. Myers, and H. Aung. Six learning barriers in end-user programming systems. In *Proc. of the IEEE Symp. on Visual Languages and Human Centric Computing*, pages 199–206, 2004.

[10] F. Mäyrä, A. Soronen, J. Vanhala, J. Mikkonen, M. Zakrzewski, I. Koskinen, and K. Kuusela. Probing a proactive home: Challenges in researching and designing everyday smart environments. *Human Technology*, 2(2), 2006.

[11] M. Mokhtari and M. Feki. User needs and usage analysis in a smart environment for people requiring assistance. *Topics in Geriatric Rehab.*, 23(1):52, 2007.

[12] A. Nijholt, R. op den Akker, and D. Heylen. Meetings and meeting modeling in smart surroundings. *Social Intelligence Design*, pages 145–158.

[13] P. Rashidi and D. Cook. Keeping the intelligent environment resident in the loop. In *Proc. of the 4th Int'l. Conf. on Intelligent Environments*, 2008.

[14] E. Rukzio, K. Leichtenstern, V. Callaghan, P. Holleis, A. Schmidt, and J. Chin. An experimental comparison of physical mobile interaction techniques: Touching, pointing and scanning. *Proc. of Ubicomp*, 2006.

[15] C. Scaffidi, M. Shaw, and B. Myers. Estimating the numbers of end users and end user programmers. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 207–214, Sept. 2005.

[16] E. Tapia, S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*, pages 158–175, 2004.

[17] K. Truong, E. Huang, and G. Abowd. CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. *Proc. of Ubicomp*, 2004.

[18] M. Weiser. The computer for the 21st century. *SIGMOBILE*, 3(3):3–11, 1999.

[19] M. Weiser, R. Gold, and J. Brown. The origins of ubiquitous computing research at PARC in the late 1980s. *IBM Systems Journal*, 38(4):693–696, 1999.