Programming Tool of Context-Aware Applications for Behavior Change

Jisoo Lee
School of Arts, Media + Engineering, Arizona State University, Tempe, AZ 85281
jisoo.lee@asu.edu

Erin Walker
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ 85281
erin.a.walker@asu.edu

Winslow Burleson
School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281
winslow.burleson@asu.edu

Eric B. Hekler
School of Nutrition & Health Promotion, Arizona State University, Phoenix, AZ 85004
ehekler@asu.edu

UbiComp'14 Adjunct, September 13-17, 2014, Seattle, WA, USA
ACM 978-1-4503-3047-3/14/09.
http://dx.doi.org/10.1145/2638728.2638735

Abstract
While users often have goals related to developing better habits (e.g., eating more healthy food, exercising more frequently), they are typically not very effective at achieving those goals. We have been developing a toolkit that provides hardware and software for users who have no programming experience to easily invent and test context-aware applications that can help them make changes in their behaviors. We have found that this toolkit needs to balance simplicity of interaction with the facilitation of a wide range of user experiences. To address this issue, we identified key temporal rule patterns from a user-generated collection of behavior change applications, and created programming elements with which users can compose applications of those patterns.

Author Keywords
Behavior change; Context-aware computing; End-user programming tools

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

Introduction
Significant work has been carried out to enhance people's behavior change in various domains,

Figure 1. Exemplar use of magnetic sensors for detecting user's object use.

leveraging Ubiquitous Computing technology. However, it is mostly provisioned as prefabricated 'solutions'. While this research is important, it ignores users' agency in determining which behaviors they want to change and how. A complementary pathway in line with the rising do-it-yourself culture would be to develop tools that can facilitate self-experimentation with behavior change strategies for creating solutions to unique personal needs. With these tools, users will be able to discover the behavior change solutions that are most effective for them.

Therefore, we have been developing a toolkit that provides hardware and software well-suited for users who have no programming experience to easily invent and test context-aware applications in an attempt to promote behavior change for a personally salient, home-based behavior (i.e., sitting/TV watching, snacking, or flossing). We focus on users' realization of the cues-to-action behavior change strategy with context-aware computing that enables just-in-time delivery of adapted information [4]. In addition to significant persuasive power by intervening at the most opportune moments, we conceive the crucial role of contextual cues in habit formation [8]. These context cues are highly idiosyncratic, and therefore require individuals to actively engage in creating the solutions as they know best about their own contexts.

Although there has been considerable research on end-user programming tools for creating context-aware applications in home environments, most tools are to support control of appliances or environmental equipment [1][2]. The provision of toolkits focused on behavior change likely involve addressing user needs that are distinct from the ones currently advanced by existing smart home control and automation systems.

One of the most salient issues encountered when developing the toolkit was to create programming tools that are simple enough for a non-computer scientist yet allow creation of a wide range of user experiences for supporting behavior change. We identified key rule patterns from a user-generated collection of behavior change applications, and determined the necessary programming elements with which users can compose applications of those patterns. In this paper, we present our approach in developing the toolkit and then describe the patterns and programming elements.

Approach of Toolkit Development

The toolkit enables rapid prototyping of rule and event-based systems that include physical sensing, data storage, and media event components [5]. This is an exemplar application that can be implemented with the proposed toolkit. A person who becomes concerned about always skipping brushing her teeth at night may have this application idea: "When I enter the bathroom between 9 and 11pm (opportune as I can immediately start brushing), I hear a silly sound effect that I set to prompt me. As I start brushing, a news podcast starts."

The toolkit includes off-the-shelf X10 sensors for detection of user's interactions with objects and spaces (Figure 1)[5]. For example, a sensor could be attached to a toothbrush for detecting the start of brushing teeth. It integrates two prompting channels: first, audio contents via location-based displays (i.e., wireless speakers), and secondly, text messages via mobile devices. The audio content includes machine speech of user-input text, and play of user-added/selected sound

Figure 2. Exemplar composition ("If TV turned on after 8 PM, a sound clip played.") of GaLLaG Strip

Examples of each pattern respectively

1. "If I keep brushing my teeth for 2 minutes, an applause sound plays"

2. "If I have not washed my hands in 10 minutes after coming home, a 'water' sound clip plays"

3. "If I brushed my teeth at three consecutive nights, my favorite songs play when I open my chocolate box."

4. "Two minutes later after an entrance door is closed, I hear music from the bathroom inviting me to washing hands"

files. We developed a first version visual programming approach, GaLLaG Strip [5], to support this process. By arranging visual elements in a linear fashion, users define if-then rules (Figure 2). This approach of starting with simple but essential logic is in line with previous research [1][7].

Key Patterns in User-Generated Rules

Although our early user studies [6] confirmed benefits of simple if-then rules proposed by the existing research (people naturally utilize them in defining applications), there were interactions in the participant-generated scenarios that cannot be implemented with the current functionality of GaLLaG Strip. Based on frequency of use and significance with respect to core behavior change techniques, we identified the following types of rules for eliciting a system response:

1. When an action continues for specific duration;
2. If another action has or has not occurred for specific duration since an action occurred;
3. If an action has or has not occurred between two absolute times;
4. When specific duration passes after an action occurs, a system response is made.

This finding teaches us that inclusion of temporal relationships is quite desirable, if not essential, in behavior change applications. Existing end-user configuration or programming tools for context-aware applications are discriminated from each other in terms of their involvement of time-related logic for conditional rules. For instance, 'Play bits' [3] does not involve any temporal logic. While 'CAMP' [7] only provides logic to define time periods (e.g., Dinner can be defined to happen "in the dining room between 7 P.M. and 9 P.M.", or "beginning at 7 P.M for 2 hours"), 'iCAP' [1] allows richer expression by further including logic for ordering (e.g., "if Tom walks in after Jane finished her dinner"). With the identified patterns, especially the first three, we conceived the need to expand the functionality proposed in the previous studies. Pattern 1 and 3 reveals the need of functionality to track the degree of performance (e.g., duration, frequency) and respond according to it, that is, enable users to involve the self-monitoring strategy. Although we focus on the application of the cues-to-action technique, it is likely that other behavior change strategies, such as self-tracking, may likely be required in the system. Second, with Pattern 2 and 3, we learned that users would like to be prompted when a behavior was missed. Thus, the second version includes logic for checking whether an activity is performed in a specified period of time.

Visual Interface Design

We determined a set of programming elements that allow composition of the identified rules. In visually representing the elements and relationships between them, we employed a block-based approach. For users' intuitive understanding, we used different interlocking shapes that constrain connections. There were a total of nine possible blocks for the condition of the rules (that is, blocks placed above the 'THEN' block in Figure 3), classified into Objects, Use of Objects, Time, Time Elapsing, and History. Time order of a rule is represented by placement of the elements, that is, an element placed above another one should occur first. Laying elements horizontally means no specification on their time order. Blocks of the Time Elapsing and History categories define time duration of specific situations, time passed after situations, and time periods to query include other blocks to describe the

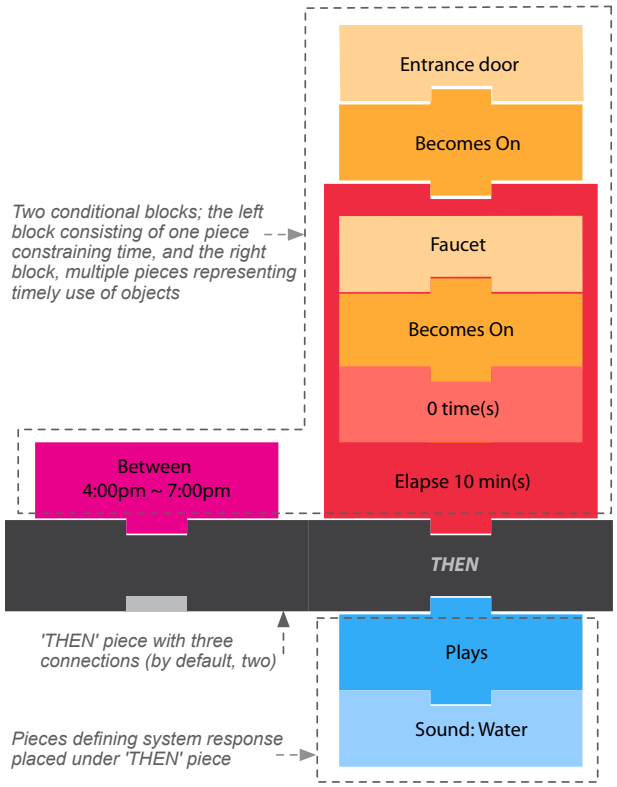

Figure 3. Composition of the Patter 2 example, "If I have not washed my hands in 10 minutes after coming home, a 'water' sound clip plays"

situations related. For example, in Figure 3, the red block ("Elapse 10 min(s)") wraps the blocks ("Faucet", "becomes on", and "0 time"), defining what should occur after "Entrance door" "Becomes On." If the user enters the house and, within 10 minutes, the faucet has not been turned on, the rule is satisfied.

Conclusion and Future Work

We developed a toolkit to support users' creation of context-aware applications more relevant to their lives and ultimately more effective at promoting behavior change. Our on-going research for optimization is to figure out functionality essential for behavior change solutions. In this paper, we presented our second iteration of the programming tool design. It extends the previous, initial tool by involving elements for the temporal relationships that we found from the user-generated scenarios. In the immediate future we will be conducting a lab study with the current paper-based prototype, to examine: (1) Can people easily understand how to use the tool, and (2) construct what they intend? This study will be followed by further design modifications, development of working prototypes, and a field study in natural environments. In addition to the effort to improve usability and expressiveness of the programming tool, we anticipate needs to enhance users' creativity and knowledge of behavior change techniques so that they can generate rich and meaningful solution ideas.

Acknowledgement

This work was supported, in part, by a Google Faculty Research Award (PI: Hekler).

References

[1] Dey, A., Sohn, T., Streng, S., & Kodama, J. (2006). iCAP: Interactive prototyping of context-aware applications. In *Pervasive Computing* (pp. 254–271).

[2] García-herranz, M., Haya, P., & Alamán, X. (2010). Towards a Ubiquitous End – User Programming System for Smart Spaces. *J. UCS*, *16(12),* 1633–1649.

[3] Humble, J., & Crabtree, A. (2003). "Playing with the Bits" User-configuration of Ubiquitous Domestic Environments. In *UbiComp 2003: Ubiquitous Computing* (pp. 256–263). Springer Berlin Heidelberg.

[4] Intille, S. S. (2006). The goal: smart people, not smart homes. In *Proc. ICOST2006: The International Conference on Smart Homes and Health Telematics* (pp. 3–6).

[5] Lee, J., Garduño, L., Walker, E., & Burleson, W. (2013). A tangible programming tool for creation of context-aware applications. In *Proc. the 2013 ACM international joint conference on Pervasive and ubiquitous computing* (pp. 391–400). ACM.

[6] Lee, J. (2013). Supporting self-experimentation of behavior change strategies. In *Proc. the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication* (pp. 361-366). ACM.

[7] Truong, K. N., Huang, E. M., & Abowd, G. D. (2004). CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. In *UbiComp 2004: Ubiquitous Computing* (pp. 143-160). Springer Berlin Heidelberg.

[8] Wood, W. and Neal, D.T. (2007). A new look at habits and the habit-goal interface. *Psychol Rev 114*, 843– 863.