SiteView: Tangibly Programming Active Environments with Predictive Visualization

Chris Beckmann, and Anind Dey

IRB-TR-03-019

June, 2003

Intel Research Berkeley

SiteView: Tangibly Programming Active Environments with Predictive Visualization

Chris Beckmann & Anind K. Dey

EECS, UC Berkeley
beckmann@cs.berkeley.edu

Intel Research, Berkeley
anind@intel-research.net

Abstract. Active environments – those with sensing and actuation capabilities – are often difficult for end users to control. We describe the design and implementation of a system, SiteView, for creating and viewing automation control rules. SiteView has an intuitive tangible interaction method for creating control rules and enhances user understanding of the system by appropriately exposing internal state. SiteView also supports users' visualization of the active environment outside the programming environment through a photographic display keyed to control rule conditions. We demonstrate SiteView's usefulness through example control rules.

1 Introduction

Research in *active environments* – those with sensing and actuation capabilities – usually involves sophisticated technology, but usable active environments need not be complicated for end users. Much automation can be achieved with simple sensing and actuation, and, indeed, commodity toolkits and protocols such as X10 exist for this purpose [9]. However, use of everyday automation tools is generally limited to hobbyists and those comfortable with traditional programming techniques. Some would argue in this case that lowering barriers for novice users is a problem that requires machine learning and heavyweight inference tools, which require little explicit input. Conversely, we argue that this can be seen as chiefly *a problem of user interface*. Specifically, commodity automation toolkits do not correspond spatially to how users interact with their environments and do not offer feedback about the internal state of the control system [9,10]. Furthermore, these toolkits lack *situated visualization* – they do not show the user the future effects of her rules at the time she is programming them.

SiteView addresses these issues by lowering the learning curve for environment automation, while maintaining enough logical expressiveness to remain useful. SiteView programs consist of rules with a simple conjunctive predicate and one or more consequent actions. Users create rules by manipulating tangible *interactors* representing sensed conditions and automated actions within a world-in-miniature (WIM) model representing the active environment. To enhance transparency, our interface

offers explicit feedback to the user during programming. It shows what control rules are applicable given the user's conditions and it provides an image of what the environment will look like under those conditions and actions specified by the user.

Our project draws inspiration from a broad base of previous work. While Mozer's neural network house investigated making home automation usable, the behavior and internal state of his machine learning system is essentially opaque to the user [7]. The Accord toolkit also supports home automation, and encourages explicit programming by direct manipulation, but it reverts to a simplified GUI as a programming environment [2]. To ease programming in SiteView, we built tangible interactors, drawing upon previous tangible programming work in Gorbet *et al.*'s Triangles and Blackwell's Media Cubes [4,3]. The world-in-miniature ties our tangible interactors to the physical environment, and originated as a technique by Stoakley *et al.* for navigation of virtual reality spaces [8]. Our environment visualization concept helps users visualize the effects of their programming decisions, and applies Abowd and Mynatt's notion of capture and access to recording recurring *environment* states, rather than the traditional sense of capturing transient *user* activities [1].

This paper describes a system for intuitive end-user programming of active environments. Section two motivates our system design with an example scenario. Section three describes our design in the context of our proof-of-concept implementation. We conclude by discussing issues in generalizing the design and by summarizing important aspects of the approach.

2 System overview

Our tangible interaction interface makes end-user programming of automated environments simpler by leveraging spatial and visual correspondences between the control interface and the automated environment, and by enabling seamless visualization of the active environment and the automation rules.

There are five main components of the interface's physical design. The first three support the act of rule creation and the last two support feedback about those rules:

- **interactors** are physical objects which logically correspond to rule conditions and automated actions;
- **world-in-miniature** is a small-scale representation of the active environment;
- **condition composer** is an area that senses and structures the user's specification of rule conditions;
- **environment display** shows what the environment will look like when a rule is activated; and,
- **rules display** interactively shows the rule as it is created and shows other rules applicable for the given set of conditions.

As an illustration, consider the following scenario. On a rainy morning, Dana finds her workspace too dark and too cold and wants to adjust the lighting and room temperature. Dana walks over to SiteView and consults the rules display, which, by de-

fault, shows the rules active in the current situation. She notes that the active lighting and temperature control rule handles weekday mornings in general, but not rainy weekday mornings in particular. Rather than manually changing the temperature and light conditions using the available thermostat and light switches, Dana decides to use SiteView so the active environment will behave appropriately now as well as the next time it is raining. She adds a new rule to handle her current situation. First, Dana places the interactors signifying **rain,** ***morning and*** **weekdays** in the appropriate slots on the condition composer (left center and right, respectively, as shown in Figure 1). The rules display (far left of Figure 1) shows all applicable control rules for those conditions, including (the currently active) one for a more general condition, weekday mornings, and the visualization display shows an image of the office similar to the office's current appearance. Next, Dana places the **light on** interactor on the portion of the world-in-miniature signifying her floor lamp. Now that Dana has specified a valid rule – both a condition and an action – the rules display shows it as an English-like sentence: if it is raining and a weekday and morning, then turn on the north lamp. Dana then sets the **thermostat** interactor to a warmer temperature and also places it in the world-in-miniature. The rules display now shows both the entire new rule, which handles light and temperature on rainy mornings, along with the original set of rules. The environment display reflects her new rule, and shows her office lit by her floor lamp on rainy mornings. SiteView then turns on the floor lamp.

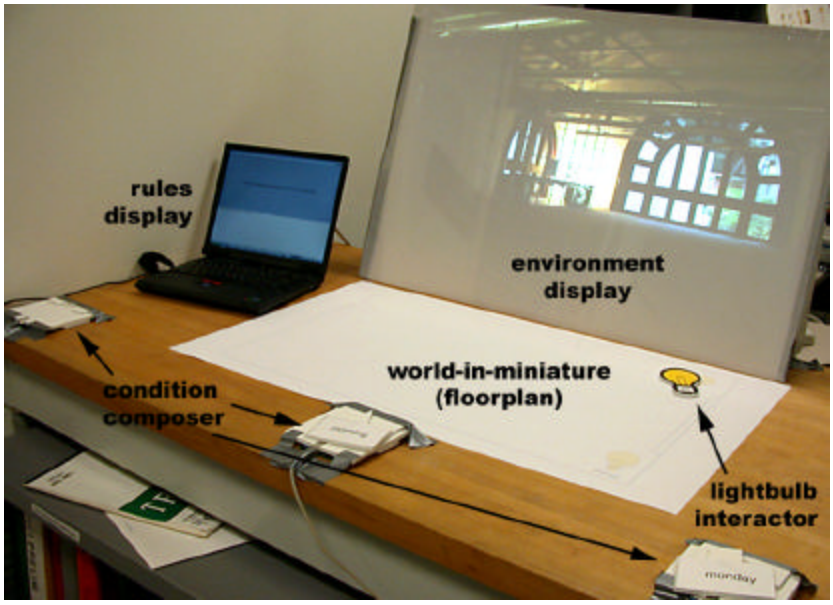

Figure 1: SiteView. The condition composer can be seen at left, center, and right; the large screen is the environment display. The laptop shows the rule display, and a lightbulb interactor can be seen at the top

left of the WIM floorplan. The configuration of interactors shown creates a rule to turn on the north lamp on rainy Monday mornings.

While Dana is making these adjustments, she thinks a little more about temperature and realizes that she finds the office uncomfortable in the afternoon, though she is not certain why. She removes the interactors from the condition composer and world-in-miniature, and places just the **afternoon** interactor into the appropriate slot on the condition composer. The rules display shows all control rules active in afternoons, including a general rule that sets afternoon temperature, and the environment display shows the appearance of the office in the afternoon. Dana notes bright sunlight on her desk in the afternoon, and wants to make her office cooler in the afternoon. She sets the thermostat interactor to an appropriate level and places it on the world-in-miniature. This does not add a new rule, but modifies the afternoon temperature control rule. The rules display continues to show the set of applicable rules, including the modified temperature control rule.

3 Proof-of-concept implementation: an active office

As a proof of concept, we implemented our interface design to provide control for a new laboratory space used by our research group. The space is very large, approximately 400m^2, and has two large windows which make areas of the room susceptible to local changes in temperature and lighting from incoming daylight. We sought to make the space more livable by implementing an interface that supports fine-grained spatial control of the temperature and lighting to compensate for local fluctuations and supports typical temporally-based home automation capabilities.

3.1 Interactors

SiteView uses tangible interactors to represent rule conditions, such as **afternoon**, and actions, such as **light on**. The benefits of tangible interfaces have been well described elsewhere [6], but are particularly suited to our system. Specifically, tangible interactors allow for collaborative and two-handed interaction, require less dexterity than traditional input, and better preserve spatial relationships between virtual objects and their real-world counterparts.

In our system, we maintain the notion of two types of tangible interactors: *tightly-coupled* and *loosely-coupled*. Tightly-coupled interactors have real-world counterparts, like thermostats, as objects or actions within the scope of the WIM, and their correspondence is particularly intuitive. Loosely-coupled interactors do not have direct counterparts within the WIM, but rather represent global states, such as day of week, or conditions outside of the scope of the WIM, such as weather.

To specify rule conditions, we created interactors representing weather, time-of-day, and day-of-week condition categories. A total of forty-two loosely-coupled condition interactors represent discrete states and ranges along each dimension: weather interactors depicted sunny, partly cloudy, overcast, and rainy states; time-of-day

interactors depicted each of the twenty-four hours of the day, as well as morning, midday, afternoon, evening, and night; and day-of-week interactors depicted each of the seven days of the week as well as weekdays and weekends. These interactors are implemented as foam-core cards, as shown in figure 2, and embedded with RFID transponders. The cards provide tactile feedback about their condition category through a unique shape on their upper edge, which also serves as a physical constraint within the condition composer.

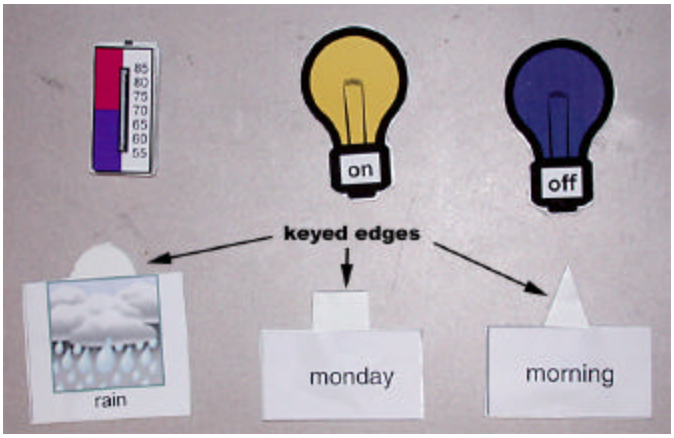

Figure 2: The interactors. The top row shows the action interactors for temperature adjustment and on/off light state. The bottom row shows the condition interactors; note the keyed upper edges.

To specify the rule actions, we built tightly-coupled interactors representing both binary light state and a thermostat, shown in Figure 2. We built four light state interactors, two for each controllable lamp in the active environment. A physical slider Phidget [5], labeled with a temperature range from 55 to 85 degrees Fahrenheit, was used for the thermostat. Both the light and thermostat interactors were tracked on the WIM by a ceiling-mounted camera.

3.2 World-in-miniature

SiteView uses the world-in-miniature as an interaction space for the user to spatially specify automation actions with the tightly-coupled interactors. The WIM is a physical artifact with a spatial and logical correspondence to the user's view of the active environment, which enhances user intuition about programming actions. In Dana's case, the WIM was a floor plan of her office, marked with the controllable lamps. While scripting languages and GUIs tend to share only a logical correspondence to the active environment, the WIM shows both active components and their spatial layout.

For our implementation, the WIM was implemented as a low-detail floorplan, showing accurate spatial orientation of the active objects in the environment as well as the walls, windows, and doorways. Since we instrumented the environment before occu-

pying the space, there were no significant non-active objects to represent. The total interaction space afforded by the WIM floorplan is approximately 100cm x 60cm.

3.3 Condition composer

The condition composer is an interaction space for specifying loosely-coupled rule conditions. Since conditions specified here do not have a physical correspondence to the real-world active environment, this area is physically separate from the WIM. Conditions may be represented as discrete tokens, such as cards, or they may be active controls, such as knobs or switches. Depending on the choice of sensing technology, the space may be large – anywhere within the range of a tether or radio link – or it may constrained to a specific actively sensed space, such as an RFID reader or camera.

We implemented the condition composer as a series of three Phidget RFID readers, masked with foam-core slots physically keyed to a particular category of condition interactor. To reduce interference between readers, the readers were separated by approximately 75cm. The slot keys were created to match the keys on the weather, time-of-day, and day-of-week interactors. The design of the condition composer allows simple conjunctive rules that accommodate only one instance of interactor per condition category. For example, Dana could specify an action for cloudy Thursdays at 8 PM, but could not specify a single rule for cloudy Thursdays *and* Fridays at 8 PM. This design constrains the user, but also serves as a guide to prevent the user from building rules the system cannot interpret.

3.4 Environment state display

The environment display shows photographs of what the active environment will look like, including ambient and lamp-provided light, for a given set of user-specified conditions (weather, day, and time). For example, Dana can now specify rainy weekdays and see an image of her office on a rainy day with her floor lamp turned on to improve the office lighting conditions. The user can also use the environment display to simply check automation settings for a particular set of conditions, including the current ones.

The environment display was physically implemented as a sheet of 60cm x 90cm Plexiglas, backed with vellum drafting paper to create a semi-translucent projection surface and placed vertically on a table surface between the condition composer readers and the world-in-miniature floorplan. We used a commodity XGA video projector to back-project environment images onto the vertical display surface. The relatively large display size serves to draw the user's attention to the environment visualization, and the resulting relative equivalence of size between the world-in-miniature and the environment display enhances the realism of the two-dimensional floorplan.

To photographically capture the environment state, we set a digital camera to take four exposures, one for each light state combination, every half hour over the course of a twenty-four hour day. Given the ubiquity and low cost of digital cameras, it is not unreasonable to expect an automated environment to have a persistent image capture capability. Naïvely, the photographic capture of the visible environment state space requires approximately five thousand unique images to represent each of the action

states in each of the condition states. This large number demonstrates the range and variety of unique rules that can be created with SiteView. The number of required images does not match the cardinality of the state space, however, since one entire condition dimension, day-of-week, is invisible, and captures done during nighttime hours are identical. To further reduce the number of unique images, we simulated environment state during various weather conditions using images taken from dawn and dusk hours.

3.5 Rule display

The rule display provides the user with explicit feedback about the internal state of the control system, which supports a more transparent user understanding of system behavior. Our system exposes state about rules currently being created, as well as those rules already controlling the system. In the former case, the rule being created by the user is displayed to the user as an English-like sentence, which gives the user immediate feedback about their manipulation of the tangible interactors. In the latter case, as the user specifies predicate conditions, the system displays the set of rules that match the set of conditions. By specifying fewer conditions, the user can display a larger set of rules.

The rules display is a Java application that runs on the laptop used to handle the RFID inputs, the vision processing, and to update the environment display. Visually, the top half of the screen displays a large, bold text sentence representing the currently specified rule. The bottom half of the screen displays in smaller text all active rules for the currently specified conditions. If a user makes an incorrect placement of one of the condition interactors, for example placing two weather interactors on the condition composer, the feedback about the currently specified rule changes to an error status message and indicates the source of the error.

5 Conclusions and future work

This paper described SiteView, a system for intuitive tangible end-user programming of active environments. We described our system design and an example interaction scenario, and detailed our proof-of-concept implementation. We conclude by discussing issues in generalizing the design and by offering directions for future work.

An initial user study of SiteView demonstrated that end users could create rules that control their environment. The tangible interface appears to be intuitive and the environmental display and rules display are useful for helping users create rules and view the effects of these rules. Overall, the system was usable for generating a variety of rules, each using one to three rule conditions, and each triggering an action that affected one or both of the lights and the thermostat settings. Additionally, the system made transparent to the user the effects of composing multiple active rules. For example, a rule that turned on the lights in the evenings was understood to be combined with another rule that set the temperature at 55 degrees on overcast weekends to turn on the lights and set the temperature to 55 on a Saturday evening. One confusion

that arose during the study was the duration of time-based rule conditions. While the use of natural words for time-of-day appeared transparent, one user was unsure if a rule that specified turning down the heat at 8 PM would still be in effect at 8:15 PM or later.

The general design of SiteView may be adapted for various user populations and granularities of control. The physical affordances of the condition composer can allow for a variable degree of scaffolding of user programming. For novices, physical constraints may be applied to the conditions to enforce the coherence of the user's constructed predicate. In the domain of conjunctive rules, for example, it would not make sense for a predicate to contain both **day=Thursday** and **day=Friday**, so the interface may physically prevent the specification of more than one day. For more expert users, the space may be less constrained and offer greater expressive power. In particular, the predicate space could be expanded to include non-conjunctive predicates, so that **day=Thursday** and **day=Friday** specifies a disjunctive relationship.

The world-in-miniature approach is also flexible in that it supports both highly reconfigurable and finely-grained interaction by varying the realism and physical dimensionality of the artifact. To strengthen correspondence with the actual environment, the environment may be represented photographically or pseudo-photographically; to avoid clutter, including only the active objects within the environment. The environment may be represented as a flat floorplan, which allows for swapping worlds-in-miniature by swapping sheets of paper, and therefore configuring multiple active environments from a single interface. Equally, it may be a represented as a three-dimensional diorama and afford users another dimension of control.

For certain conditions and actions, it may be argued visualization of an environment is not appropriate. Granted, non-visual environmental conditions, such as day-of-week, likely has no effect on the appearance of an environment, but since the user expects no visible state, this merely simplifies the visualization task. For non-visual actions, it is difficult to imagine a natural photographic visualization of the controlled temperature of an environment, for example. While temperature has a direct impact on experience of the environment, it is poorly represented by the visualization. In this case, we argue that the visualization of intermediate factors, such as ambient outdoor light, may assist users in their control task.

References

1. Abowd, G.D. and Mynatt, E.D. "Charting past, present and future research in ubiquitous computing". *ACM Trans. on Computer-Human Interaction*, 7(1):29-58. (2000).
2. Åkesson, K-P. *et al.* "A toolkit for user re-configuration of ubiquitous domestic environments". In *Companion to Proceedings of UIST 2002.* (2002).
3. Blackwell, A.F. and Hague, R. "AutoHAN: An architecture for programming the home". In *Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments*, pp. 150-157. (2001).
4. Gorbet, M.G. *et al.* "Triangles: Tangible interfaces for manipulation and exploration of digital information topography". In *Proceedings of CHI '98*. (1998). pp. 49-56.

5. Greenberg, S. and Fitchett, C. “Phidgets: Easy development of physical interfaces through physical widgets”. *In Proceedings of UIST 2001.* (2001). pp. 209-218
6. Ishii, H., and Ullmer, B. “Tangible bits: towards seamless interfaces between people, bits and atoms”. In *Proceedings of CHI'97.* (1997). pp. 234-241.
7. Mozer, M.C. “The Neural Network House: An environment that adapts to its inhabitants”. In *Proceedings of the AAAI Symposium on Intelligent Environments* (1998). pp. 110-114.
8. Stoakley, R. *et al.* “Virtual reality on a WIM: Interactive worlds in miniature”. In *Proceedings of CHI '95.* (1995). pp. 265-272.
9. Smarthome X10 Kit. http://www.smarthome.com/x10map.html
10. Home Director. http://www.homedirector.com/