Context-Aware Computing Support for the Educationally Disadvantaged

Timothy Y. Sohn
University of California, Berkeley
tsohn@cs.berkeley.edu

Abstract

The educationally disadvantaged people in context-aware computing are those without any programming experience, and are unable to build context-aware applications with the current level of available prototyping support. iCAP is a system that assists this group of non-programmers in visually designing context-aware applications without writing any code. Individuals are then able to rapidly prototype their application in a simulated or real context-aware environment. Through iCAP, programming context-aware computing is made intuitive and viable for the educationally disadvantaged.

1. Introduction

Environments contain *context* information that can be sensed and used in programming applications. This information is typically described in the categories of location, identity, activity, and time. Context-aware computing involves the sensing of available resources to provide appropriate information and services depending on the state of each element in an application's context. In recent years, futuristic computing environments that utilize contextual information have sparked interest within the general public to experience these new spaces. Recent movies such as Minority Report, demonstrate the novelty of ubiquitous computing, and leave the audience in amazement, yet curious about how they could ever program off-the-desktop applications. Over the past several years, there has been an increased effort and interest in building and deploying context-aware applications, hoping to bring the movie-like environments to our present reality. However, although some of these applications have been developed, there is still a lack of programming support to rapidly develop them. Currently, to develop a context-aware application, developers and end users alike are required to either build the application from scratch involving direct interaction with hardware sensors and devices, or to use an enabling toolkit [1]. Even with low-level toolkit support for acquiring context, individuals are still required to write large amounts of code to develop simple sensor-rich applications. This obviously leaves the general public, without any programming experience, at a disadvantage to prototype context-aware applications. In addition, because of the novelty of context-aware computing, the hardware to sense environmental information is not easily accessible. Thus, many individuals who are without programming experience, and may come from minority and rural areas that are economically challenged are never exposed to this new era of ubiquitous computing.

The educationally disadvantaged people in context-aware computing are those who do not have any programming experience, particularly in the domain of context-aware computing. Since the only programming support available to prototype context-aware applications are low-level toolkits, the amount of code an individual without any programming experience would need to write is absolutely daunting and infeasible. Therefore, in order to provide power and control over context-aware environment to these individuals, it is imperative to bridge the gap between the user and low-level toolkits.

The interactive Context-aware Application Prototyper (iCAP) is a system aimed at lowering barriers for non-programmers to build context-aware applications *without requiring them to write any code*. iCAP is a visual environment that acts as the intermediate layer between low-level toolkits and users, allowing users to prototype and deploy their application to a real context-aware environment, or to simulate the application within the iCAP system. The simulation feature is important for people in minority and rural areas who are unable to easily obtain the hardware to instrument a ubiquitous computing environment. By building an application with iCAP and simulating it, users are able to gain confidence in their programming abilities, and can easily transition to deploying their application when a real context-aware environment becomes available.

2. The iCAP Interface

iCAP provides an intuitive visual environment that takes advantage of human spatial reasoning skills [2], making it simple for those without any programming experience to prototype context-aware applications. The system uses a lightweight, informal visual interface that gives users the familiar feel of sketching and rapid prototyping as if designing on paper. This is especially useful for non-programmers who may be intimidated by

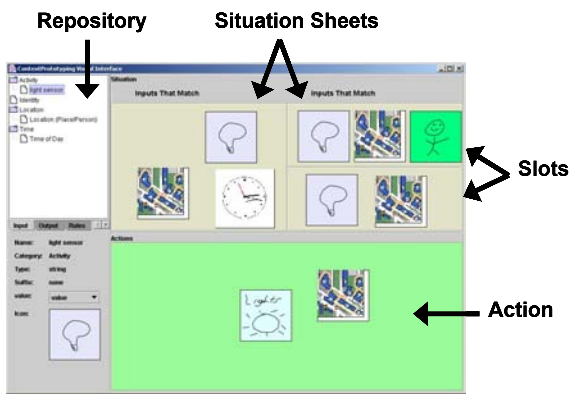

Figure 1 The iCAP user interface with a rule that has two situation sheets, where one is split into two slots.

technology and find themselves insecure in using a fancy formal interface.

The iCAP interface has one window with two main areas (Figure 1). On the left is a tabbed window that is the repository for the user-defined inputs, outputs, and rules. All of these components, except rules, are associated with graphical icons that can be dragged into the center area, and then used to construct a conditional rule statement.

The center area contains two sections, one for the situations (IF) and the other for actions (THEN). The center area uses the Pane and Myers' matching scheme to support users in visually specifying the Boolean logic of each rule [3]. The matching scheme was originally tested and validated with both children and adult non-programmers. To implement the matching scheme, our system uses two important metaphors, "sheets" and "slots". All components on a single sheet are related to each other by a conjunction. However, each sheet is constrained by only allowing one location object. We allow for sheets to be split into multiple slots, allowing each slot to have its own location object, and having all slots on a single sheet related to each other by a conjunction. In essence, this allows for expressions such as [(location1 AND b) AND (location2 AND d AND e)]. Users can also add multiple sheets, which are related to each other by a disjunction. Disjunction on the action side is rare, thus we allow for multiple action slots, but only a single sheet.

3. Interaction

Interaction with the iCAP visual interface consists of three simple steps. First, the user sketches numerous inputs and outputs. This includes defining different locations and people in the context-aware world. Then these elements are dragged and dropped into the center area to construct rule-based conditions. Finally the entire set of rules is prototyped and simulated (or deployed to a context infrastructure) using the prototyping mode in the iCAP system.

An example application that can be easily built using iCAP is shown in Figure 1. The rule being prototyped is "if the lights are on in the kitchen past 10pm, or someone is in the bedroom with the lights on and the lights are also on in the kitchen, then turn off the lights in the kitchen." The necessary components to build this application are a bedroom location, kitchen location, and a light bulb. The user can define each of these components through the repository of user-defined elements. The appropriate components are then dragged into the center area, and arranged appropriately on the sheets. The user then identifies the desired trigger value for each input (*i.e.* set the time to be 10pm), and the desired action for each output (*i.e.* set the light bulb to turn off). The rule is then setup to turn off the lights, when the lights are on in the kitchen past 10pm, or someone has the lights on in the bedroom.

4. Conclusions

iCAP provides support for non-programmers to explore the domain of context-aware computing through an intuitive visual interface. Without having to write any code, individuals are able to program their environments and experience the richness of ubiquitous computing. iCAP is equipped with features for advanced applications that take advantage of relationship-based and personalization-based features. As the novice programmer becomes comfortable with simple context-aware applications, he can continue to use iCAP and easily create more advanced applications while building experience and confidence in dealing with technology. The next steps with iCAP include further validation of the tool through formal user testing, and improving the expressiveness of the tool with respect to the variety of applications that can be built with it.

5. References

[1] A.K. Dey, D. Salber, and G.D. Abowd. "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications." Human-Computer Interaction Journal, 16(2-4), pp. 97-166. 2001.

[2] S.K. Chang. *Principles on Visual Programming Systems*, Prentice Hall, New Jersey, 1990.

[3] J.F. Pane and B.A. Myers, "Tabular and Textual Methods for Selecting Objects from a Group." In Proceedings of International Symposium on Visual Languages, pp. 157-164. 2000.