iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications

Timothy Y. Sohn
EECS Department, Computer Science Division
University of California at Berkeley
Berkeley, CA 94720 USA
tsohn@cs.berkeley.edu

Anind K. Dey
Intel Research, Berkeley
Intel Corporation
Berkeley, CA 94704 USA
anind@intel-research.net

ABSTRACT
iCAP is a system that assists users in prototyping context-aware applications. iCAP supports sketching for creating input and output devices, and using these devices to design interaction rules, which can be prototyped in a simulated or real context-aware environment. We were motivated to build our system by the lack of tools currently available for developing rich sensor-based applications. We iterated on the design of our system using paper prototypes and obtained feedback from fellow researchers, to develop a robust system for prototyping context-aware applications.

Keywords
Informal prototyping, context-aware computing

INTRODUCTION
The emergence of context-aware applications, those that take into account their context of use, has shown the ability for rich interaction with the surrounding environment. However, although some of these applications have been developed, the proliferation of context-aware applications is inhibited by the lack of programming support to rapidly develop them. Currently, to develop a context-aware application, developers are required to either design their own application from scratch, directly interacting with devices, or use a toolkit [1]. However, even with low-level toolkit support for acquiring context, experienced developers are still required to write a large amount of code to develop simple applications. A context-aware application typically consists of an infrastructure to capture context and rules governing how the application should respond to changes in this context.

iCAP is the intermediate layer between low-level toolkits and users, providing a powerful tool for developing interesting, complex context-aware applications, while allowing developers to prototype applications *without writing any code*. iCAP is an informal pen-based tool that allows users to quickly define input devices that collect context and output devices that support response, create application rules with them, and test the rules by interacting with the devices in a *run mode*. The behavior of created devices can either be simulated by this tool, or mapped to actual devices.

We built iCAP using the Java 2 SDK version 1.4, on top of SATIN [2], a toolkit for building informal pen-based interaction systems.

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

The center area contains the two elements of a conditional rule statement, which is inherent within context-aware applications. An example rule is: *if John is in the office after 5pm and the temperature is less than 50 degrees or*

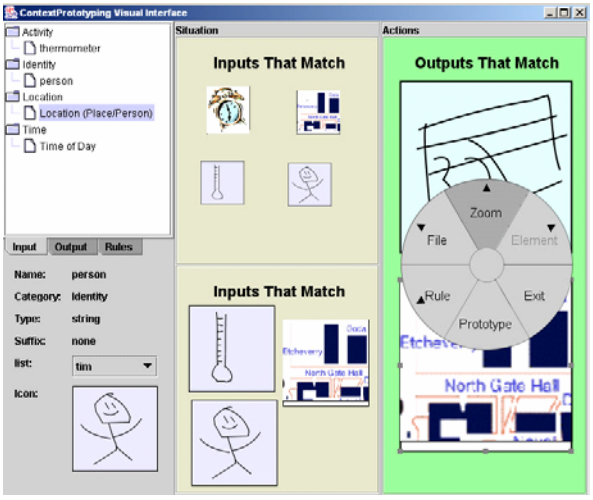

Figure 1. The iCAP user interface with an example rule that uses two input sheets

CHI 2003, April 5-10,2003, Ft. Lauderdale, Florida, USA.
ACM 1-58113-630-7/03/0004.

if Jane is in the bedroom and the temperature is between 30 and 60 degrees, turn on the heater in the house (Figure 1). The left side represents the ''if'' portion of the rule conditional, and can be split into one or more ''sheets''. Inputs on a single sheet are related by a conjunction and multiple sheets are related by a disjunction. The right side of this area represents the ''then'' portion of the rule condition. Disjunction amongst different outputs is rare, thus only a single output sheet is currently supported. We implemented Pane and Myers' matching scheme to allow users to visually specify the Boolean logic of each rule [3].

Instead of traditional pull-down menus for executing commands, we use pie menus to better support pen interaction. In addition, we also support gestures for issuing common commands such as cut, copy, and paste of inputs and outputs.

INTERACTION

iCAP involves specifying inputs and outputs, using these elements to construct application rules, and then testing the entire set of rules in a run mode.

Creating Inputs and Outputs

Each input and output component in iCAP is associated with a graphical icon. These icons are sketches drawn by the user upon creation of each component. Each icon is colored differently depending on whether it is an input or output device.

The repository window pie menu supports creation of inputs. Each input contains a suffix (*e.g.* degrees Celsius for temperature), type (*e.g.* integer, string), and four categories or primary types of context: Activity, Identity, Location, and Time. An input's potential values can be provided as a range or list.

Outputs are created in the same manner as inputs, however contain different parameters to specify. Each output is either a binary or a gradient device. By default, the number of levels in a gradient device is between 1 and 10 inclusive. In addition, there are five categories an output device is associated with corresponding to the five human senses: Sight, Sound, Smell, Taste, and Touch.

Constructing Rules

Rules are constructed by dragging and dropping inputs and outputs onto the ''if'' and ''then'' sheets of each rule. For example, if the user were interested in a temperature sensor, he would define a temperature input, and drag the corresponding icon onto the respective sheet. After dragging each corresponding icon, the user needs to setup certain parameters, or *conditions*, governing the behavior of the input. Using our temperature sensor, the user may want to know when the temperature is less than 50 degrees, or possibly between 30 and 60 degrees. We allow the user to specify a conjunction of up to three conditions using the following operators: less than, less than equal, greater than, equal, not equal. Multiple condition sets can be defined, and are all related by a disjunction.

Evaluating the Application

After a number of rules have been defined, the entire rule set can be tested using the iCAP engine in run mode. The engine can either be set to simulate the context-aware environment, or be used in conjunction with a real context-aware environment [1]. Users can interact with the engine to change the value of defined inputs, and evaluate the behavior of the rules being tested. With the engine, users are able to quickly design and test their applications, without having to create an entire infrastructure for collecting or simulating context and without writing any code.

RELATED WORK

iCAP has been inspired by previous work in ubiquitous computing applications, specifically those that involve the development of rule based conditions. Some of these include AgentSheets [5] and Stick-e notes [4]. These tools are designed for building applications, while iCAP is focused on helping developers rapidly prototype, test and iterate on their context-aware applications.

FUTURE DIRECTIONS

While we have received informal feedback from local designers of context-aware systems, we are planning to conduct a more formal study of our iCAP with a number of real users to see what features are used, and how to improve interaction with the system. Our goal is to enable both designers and end-users with the ability to create and modify context-aware applications, giving them the power that only programmers enjoy today.

REFERENCES

1. Dey, A.K., Salber, D. and Abowd, G.D. ''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. Hong, J.I. and Landay, J.A. ''SATIN: A Toolkit for Informal Ink-based Applications.'' In Proceedings of User Interface and Software Technology, pp. 63-72. 2000.
3. Pane, J.F. and Myers, B.A. ''Tabular and Textual Methods for Selecting Objects from a Group.'' In Proceedings of International Symposium on Visual Languages, pp. 157-164. 2000.
4. Pascoe, J. ''The Stick-e Note Architecture: Extending the Interface Beyond the User.'' In Proceedings of Intelligent User Interfaces, pp. 261-264. 1997.
5. Repenning, A. ''Creating User Interfaces with Agentsheets.'' In Proceedings of Symposium on Applied Computing, pp. 190-196. 1991.