Customizing User Interaction in Smart Phones

Article *in* IEEE Pervasive Computing · August 2006

DOI: 10.1109/MPRV.2006.49 · Source: IEEE Xplore

8 authors, including:

Tapani Rantakokko
Finwe Ltd.
14 PUBLICATIONS **107** CITATIONS
SEE PROFILE

Juha Kela
Finwe Ltd.
23 PUBLICATIONS **1,166** CITATIONS
SEE PROFILE

Jani Mäntyjärvi
VTT Technical Research Centre of Finland
97 PUBLICATIONS **5,258** CITATIONS
SEE PROFILE

Jonna Häkkilä
University of Lapland
317 PUBLICATIONS **4,196** CITATIONS
SEE PROFILE

Customizing User Interaction in Smart Phones

Given individual variations in mobile application use, preprogramming devices to meet user demands is difficult. A context-based, device-centric tool prototype lets users easily customize smart phones, offering low learning costs and high usability.

Panu Korpipää, Esko-Juhani Malm, Tapani Rantakokko, Vesa Kyllönen, Juha Kela, and Jani Mäntyjärvi
Technical Research Centre of Finland

Jonna Häkkilä
Nokia Multimedia

Ilkka Känsälä
Avantone

Smart phones combine mobile phone capabilities with a versatile computing platform that accepts third-party software. As with the PC, the smart phone's emergence has led to a rapidly increasing number of available applications, along with new input sources for application interaction. Among the input sources are various sensors and readers for visual and radio frequency identification (RFID) tags that are wirelessly connected to or embedded in the smart phones. Such devices facilitate novel and multimodal interaction methods, including pointing, free-form gestures, and implicit, context-based control. In addition to controlling the mobile applications, a smart phone can act as a central controller for interacting with external appliances through Bluetooth, messaging, and Internet Protocol (IP) networking, for example. Interaction convergence has already begun; most people will soon use a single device, with various modalities, for an increasing number of control tasks.

The problem here is that individuals vary widely in how they use mobile applications in different situations, and a person's preferences about such things can change over time. So, it's difficult to program mobile devices in the design stage to meet different user demands in various usage situations. To contend with this, we need a tool that lets end users easily and efficiently customize their mobile devices. Existing research in this area typically takes an infrastructure-centric (or *smart space*) perspective, relying on the environment to monitor the context and control the device.

In contrast, we've developed a context-aware solution for existing mobile devices and their applications that people can use independent of external infrastructure. The solution is hence applicable in general mobile usage rather than being restricted to a specific room or space. Our primary goal is to offer a device-centric approach with a broad scope of customizable functionality, while maintaining the low learning costs that end-user development advocates.[1] To that end, our solution uses new input modalities, including real-time pattern recognition of acceleration-based gestures, a visual tag reader, and an RFID tag read/write accessory. Users can customize the modalities to activate any available mobile device functions.

Framework overview

We had five general usability requirements for our tool: it had to be easy to learn, effective, efficient, and satisfying, and give users direct control when defining device functionality. To help meet these goals, we approached customization from a context-aware viewpoint. We define interaction inputs as *contexts* that users can connect to application actions by defining context-action rules. Supporting generic man-

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/06/$20.00 © 2006 IEEE

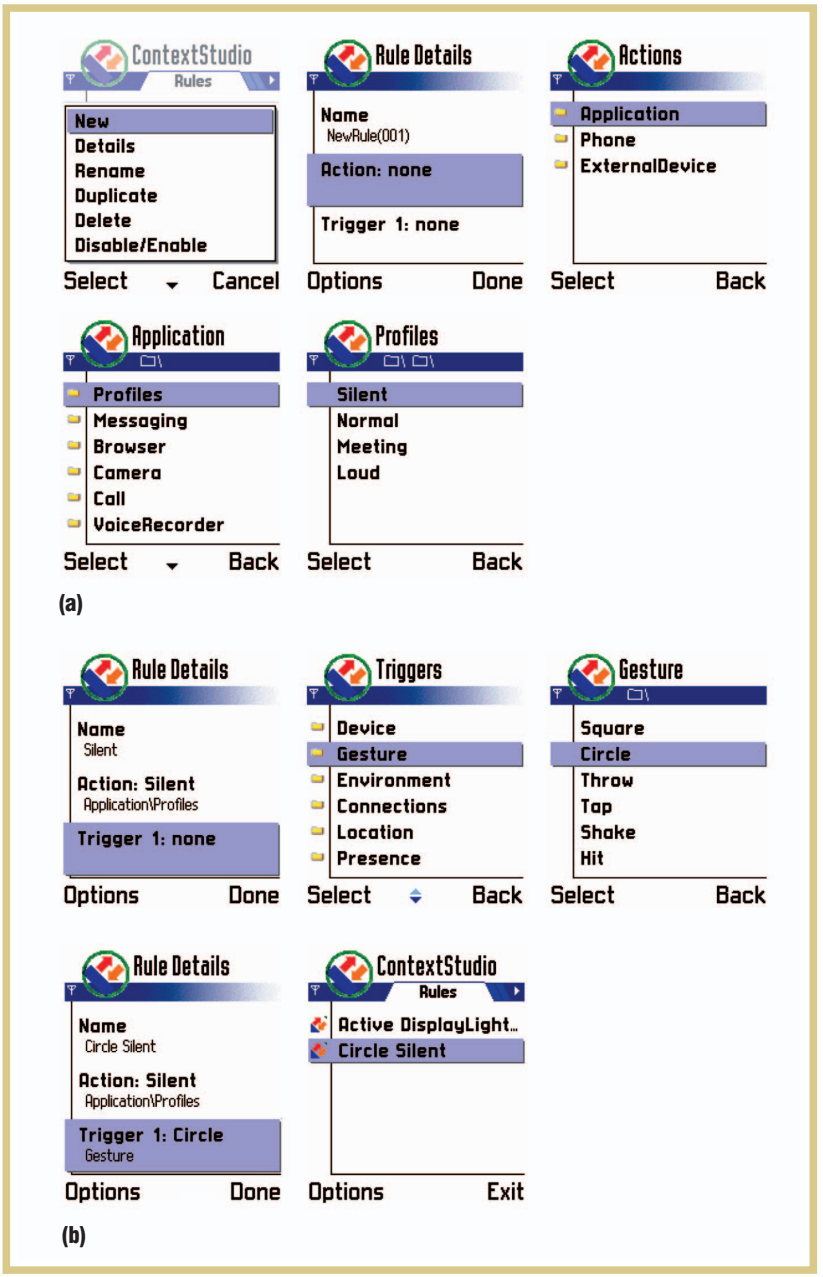


Figure 1. Rule creation on a Symbian Series 60 user interface. (a) The user chooses a new rule from the main view and selects an action element, (b) then selects a context trigger element for the rule. Navigation proceeds from left to right and ends at the main view, with the new rule highlighted in the rule list.

agement of interaction inputs and systematic rule-based mobile application control requires an enabling software framework.

Jani Mäntyjärvi and his colleagues originally introduced the Context Studio concept as a tool for customizing context-aware applications.[2] We modified and further developed the concept into a tool for small-screen mobile devices and introduced a vocabulary model to automatically generate GUI views.[3] However, the tool and GUI are not enough for complete operation. To enable real, online use of customized features in a smart phone, we utilize the blackboard-based context framework.[4]

Customization tool GUI

Our principal idea is that, instead of implementing rigid context-aware features at design time, we give users a set of available contexts and actions and let them decide whether and how to use them. Users customize their phones through a GUI by specifying context-action rules. Rule conditions are called *triggers*, which can be any event or state. An *action* is any application, function, or event that the phone activates when a set of triggers are fulfilled. Actions can also belong to external devices. We adopted Context Exchange Protocol (CEP) syntax[5,6] to formally represent the rules that the tool generates.

As figure 1 shows, the tool generates user interface views based on the rule model and the context and action vocabularies.[3] This example shows the vocabulary on a Symbian Series 60 user interface, but it can be generalized to other platforms. The user interface represents context and action types as directories, and values correspond to files. This enables straightforward user interface updates when new actions and contexts become available.

As the figure shows, selecting rule elements resembles navigating a directory tree hierarchy. In figure 1a, the user selects an action for a new rule by navigating through the Application\Profiles action type, and selects the Silent action value. In figure 1b, the user selects a context trigger for the rule by navigating through the Gesture context type, and selects the Circle context value. The Rule Details screen (figure 1b) shows the complete rule, and the rule name is generated accordingly. Once the user selects Done, Context Studio shows the new active rule name in the rule list and generates the rule CEP script.

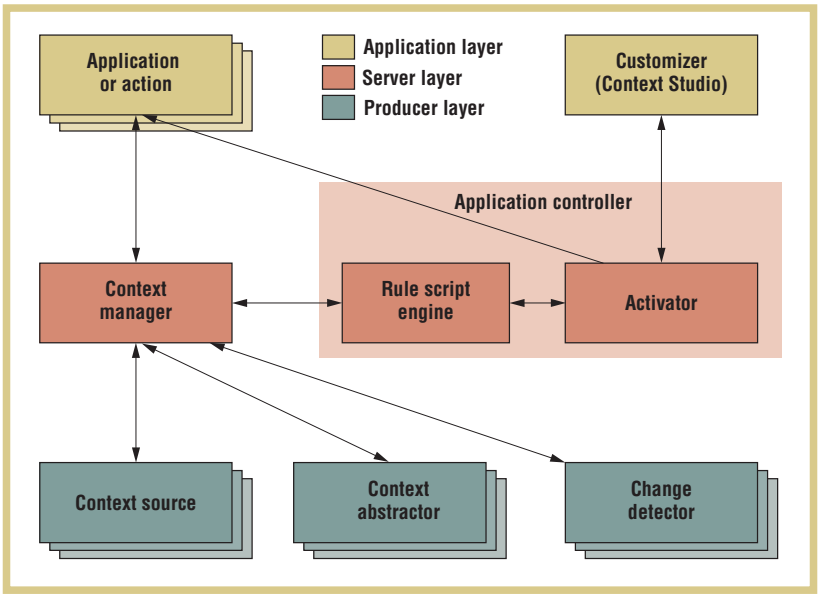


Figure 2. An overview of the extended context framework. Although applications can use the context manager directly, the application controller and customizer components let users add context awareness to existing applications without programming.

Context framework for customization

The blackboard-based context framework[4] enables the actual use of customized smart phone features. From the context-management viewpoint, Context Studio acts as a graphical editor for generating and reading rules. Rules can be created by the primary user, exchanged among users, or provided by third parties. Once a user has created the desired context-action functionality, the context framework controls applications accordingly, on the basis of context events.

To perform the customized application control inference, we extended the framework with an application controller (see figure 2). So, the framework completely separates context management from the application code, enabling context-based features without changing existing applications.

As figure 2 shows, the application controller consists of two parts: the rule script engine[5,6] and the activator. The rule-based approach for application control inference doesn't incorporate mechanisms for handling uncertainty. To eliminate uncertainty, the framework uses producer-layer components—the context sources and context abstractors (see figure 2). This prevents sensor signal uncertainty from being transmitted up to the application controller (and thus to the user).

For example, the user can train a phone in free-form gestures by repeating the gestures two to three times. This is possible because the context abstractor can learn hidden Markov models from the accelerometers' noisy gesture signals. If the user doesn't repeat the gestures similarly enough when training them, however, the learned HMMs might be incomplete. When the user subsequently makes such a gesture, the producer layer components must reduce the effects of signal noise and gesture model incompleteness and produce a discrete classification result. The application controller then uses the result to make a discrete application action, such as opening a browser bookmark. In accelerometer-based, free-form gesture control, the HMM recognition accuracy is excellent and researchers have used it successfully for such things as design environment control.[7]

Finally, the context framework lets users define multimodal control of their smart phones. The framework does this by simultaneously managing both implicit and explicit input events as contexts. Although the framework also lets users customize individual context sources, that functionality is beyond this discussion's scope.

Prototype

To evaluate our rule-based, end-user customization approach, we developed a prototype consisting of the Context Studio (figure 1), the context framework (figure 2), accelerometers, and the RFID tag reader/writer accessory (see figure 3b). We implemented the prototype for Symbian Series 60 devices and successfully tested it on five different smart phone models.

To provide real context triggers for the interaction customization, we implemented several context sources. These provide context information such as accelerometer-based, freely trainable gestures and other movement abstractions, including orientation and activity level, pointing with RFID tags, cellular-network-based location, time, Bluetooth devices, and several device platform events. These events include keyboard and display inputs; battery and network strength; charger status; profile setting; foreground application activity; and keypad lock. Users can select from multiple values for each context type provided by the context sources.

We also implemented several application action plug-ins to the activator. Among the application actions and system events are functions such as call, messaging, camera, profiles, browser, display functions, keypad, and joystick. External device actions include the Nokia observation camera functions. We described context triggers and application actions according to the vocabu-

Figure 3. Pointing input. (a) After creating the rule, the device owner uses (b) a tag reader/writer to touch the RFID tag (the green chip) and call (c) the pictured person. The tag reader uses Bluetooth to connect to the phone.

lary model. In all, we implemented and made available for customization more than 100 triggers and actions.

Framework operation example

We'll illustrate prototype functions using the following example. Let's say that a user, Mary, uses Context Studio to create the simple active rule in figure 1b: "If Gesture is Circle, then Application\Profiles should be Silent." When the user presses Done in the user interface, Context Studio generates a CEP rule script and informs the activator of a new active rule (figure 2). Activator stores the new script and adds (subscribes) it to the rule script engine, making the rule functional.

When the user makes the Circle gesture, the corresponding context source reads data from accelerometers and executes a recognition using the HMM representation of previously trained free-form gestures. Context source represents the recognition result as a context vocabulary instance. The context source encodes that instance as a context object and adds it to the context manager blackboard.

After the rule script engine receives the rule, it subscribes to the context manager so that it will be informed about rule-related gesture events. When the context manager sends a new gesture event, the rule script engine evaluates the rule. If the incoming event value matches the rule condition—in this case, Circle—the rule script engine indicates the activator, which changes Application\Profiles to Silent.

Rule-processing issues

We tested the prototype with about 30 different rules. Even when multiple rules function simultaneously, we noticed no delays in the framework's operation. When we defined multiple actions for the same trigger, rules executed in the order of creation. If two rules conflicted—that is, used the same trigger to activate two or more contradictory actions—actions were executed on the basis of rule creation order.

Although we considered automated conflict prevention, this would decrease system flexibility and increase complexity because it requires defining all possible conflicting actions for each action. Another option is to let users select from a list of actions defined for the same trigger. This is infeasible for our purposes, however, because actions must be automatic once users define the rules; system interruptions would disturb the interaction. Finally, we could let users prioritize their rules. However, users would still have the responsibility to define nonconflicting rules and their execution order. We avoid possible deadlock situations by not allowing actions to function as triggers.

External-device control

When we built the prototype, our main goal was to study how users might customize emerging interaction modalities for smart phone applications. We also showed that the same approach can be applied in relation to external devices, demonstrating control of an observation camera with an short-message-service message command interface.

Typically, for the camera to take an image and return it to the phone, users must write an SMS message containing a command and send it to the camera. With our prototype, users can customize an RFID-tag-based action ExternalDevice\ObservationCamera TakePicture. Then, when the user touches a trigger RFID tag with the tag reader, the activator sends a corresponding SMS message command to the camera. This streamlines the user interaction for controlling the observation camera.

For more general external-device control settings, smart phones can use the universal-plug-and-play (www.upnp.org) framework to control devices such as home appliances. To enable this, users connect their home appliances as UPnP devices to the smart phone, which can then

- act as a UPnP control point and perform a service search for appropriate home appliance functions,

- use the UPnP protocol to receive available actions from the home appliances and execute commands to them, and
- use Bluetooth to establish an IP connection to the UPnP server.

Although we didn't demonstrate a UPnP extension with this prototype, we can extrapolate on the functionality. With our context framework, the activator would connect to the phone's UPnP control point interface to perform actions triggered by context events. Users could, for example, turn on a TV set by making a gesture with the phone.

Emerging input modalities offer additional ways of interacting with smart phones—that is, the interaction becomes increasingly multimodal.

Using customizable input modalities

Emerging input modalities offer additional ways of interacting with smart phones—that is, the interaction becomes increasingly multimodal. In addition to basic keyboard input, existing, widely adopted input modalities include speech, joystick, or touch-sensitive displays. With our prototype, we experimented with pointing, free-form gestures, and implicit control.

Pointing

With pointing, the device identifies and selects different objects in the environment and retrieves information from those objects using RFID or visual tags.[8] As figure 3 shows, we applied a separate RFID tag reader/writer accessory box for our prototype.

Customizable multimodal interaction facilitates personalized and possibly more efficient device interaction. Among the new modalities are shortcuts or "soft keys" that decrease the click distance—the number of user operations that a task requires. To call a person, for instance, the average click distance is five or more, depending on the phone model, keypad lock state, and the target person's position in the user's phonebook. When users make calls by touching an RFID tag, the click distance is one (assuming the tag is available, of course). Such easy operability might be particularly significant for elderly people, for example, who could quickly contact relatives by touching the customized device to a picture containing a tag. With the tag read/write accessory, users can write RFID tags for any context and connect them through Context Studio to any available action. In the future, mobile phone manufacturers are likely to increasingly embed RFID tag readers directly into phones. Already, visual tag reader hardware (cameras) exist in many phone models, ready for pointing-based applications.

Free-form gestures

Users can also employ gestures to decrease click distance. Here, gestures refer to user hand movements that the phone reads using acceleration sensors. Because we use HMMs to model the acceleration-based gestures, users can train and use gestures of any form. They can also exchange gestures with other users or acquire gestures from third parties. People might use gestures as shortcuts for opening applications such as messaging, camera, and calendar, as well as to open bookmarks in a browser, send messages, play games, and so on. Users can also use gestures to easily control profiles and keypad lock.

Implicit control

Implicit interaction refers to the control mode when the user isn't directly giving commands. A simple example is a rule such as, "If device orientation is display up and device is active, turn display light on." The smart phone's accelerometers can detect the phone's activity and orientation. Another example is a rule stating, "When location is home and device charger is charging, save new images into the image album through a Bluetooth connection." In this case, the network ID context source detects coarse location from the cellular network ID codes, and the charger context source relays changes in charger status to the context manager.

Evaluation

To evaluate the usability of our rule-based customization approach, we arranged a user test for 10 participants (none of whom worked in the mobile phone industry). Four participants were males and six were females. They tested the implemented prototype on the Series 60 smart phone. During development, we tested the tool's user interface with two iterations of paper prototypes and improved it accordingly.[3] To design the user interface, we applied Jakob Nielsen and Robert Mack's usability heuristics[9] and the Nokia Series 60 style guide. This testing during development was highly valuable because it reduced the need for corrections after implementation.

Our usability study's main goals were to verify the feasibility of both the rule-based customization approach and the tool's user interface, and to gain feedback on the new interaction modalities. The test consisted of five scenarios.

After briefly introducing the tool, we gave all five scenarios to participants and asked them to use the tool accordingly. We offered no further advice. As with the applications we described earlier, the scenarios only implicitly referred to rule making. The framework's context and action vocabularies, transformed into a directory hierarchy, consisted of over 100 instances that participants navigated to select the rules' elements. We recorded a video of the tests, using a screen camera. We also encouraged participants to think aloud during the test, and recorded their comments. After completing the tasks, the participants completed a questionnaire charting their opinions.

Because all participants completed all given tasks, we consider our approach effective. We also rated our results according to three other criteria: ease of learning, efficiency, and satisfaction.

Ease of learning

Nine participants understood the idea of constructing context-action rules at the start of the first task. One participant needed guidance in how to construct a rule in that task; four asked for either confirmation or correction concerning their understanding of the trigger and action concepts. After the first task, all 10 completed the tasks without further advice on rule construction.

Efficiency

Participants completed the tasks with no errors. However, the context trigger and action directories contained over 100 elements that were unknown to users beforehand. So, they initially had to perform searches to find the desired elements, and they occasionally asked us questions related to the directory structure. The navigation efficiency was the worst in scenario 3, where nine participants initially looked for the required trigger in the wrong folder. By comparison, only three participants used the Back key at some point during scenario 2, and only one used it in scenario 4. Although we iterated the vocabulary according to the user feedback from our two paper prototype tests, our results indicate that the vocabulary requires further additional work because it's crucial for user interface navigation efficiency.

Satisfaction

In the written feedback, eight participants said that they would benefit from using the tool. When we asked the participants to rate the tool's general usability on a scale from 1 (worst) to 5 (best), the average was 3.7, with a standard deviation of 0.9.

We also collected user feedback on the interaction modalities. In three scenarios, the participants performed real interaction with the prototype after they'd used the tool to specify the rules. We asked the participants which test feature they liked the most. All answered, and two participants named two preferred features, for a total of 12. Five participants mentioned gesture control, three mentioned implicit interaction based on sensed contexts, and four liked using a phone to control external devices. No one mentioned RFID-tag-based pointing, which might be either because it currently requires the separate tag reader or because of the difficulty in quickly perceiving the possibilities of a completely unknown interaction concept. In this sense, gestures are advantageous because they're innately natural to human communication. As for implicit control, the users had different opinions about the subjective contexts, such as Environment\Sound\Intensity Loud. This suggests that we should either let users define the subjective contexts, omit those contexts, or at least make the description of the subjective context value's meaning readily accessible.

Discussion and further work

An important challenge in end-user customization systems is to give users an experience of control. In other words, the actual system functionality should exactly match the functionality that the users want to create as a result of the customization process.

Existing approaches

As the "Related Work in End-User Customization" sidebar describes, among the proposed approaches to context-based customization is that of Anind Dey and his colleagues, who propose modeling contexts on the basis of examples.[10] This is feasible when an example contains a single, user-identified context type. When an example contains multiple context types, this programming-by-demonstration approach can lead to unintended functionality if users can't control which contexts are relevant for their intended actions. Khai Truong, Elaine Huang, and Gregory Abowd propose to provide users with a set of words that they can arrange for a description, which the system then translates into functionality.[11] Because the system's set of parameters is different from the words available to the user, however, the user-created descriptions can be ambiguous and don't always result in the intended functionality.

Rule-based modeling

In our rule-based approach, users define each rule's condition and action individually and explicitly using type-value pairs. As our evaluation results show, this approach yields satisfactory usability and user control. The test participants understood well the idea of defining device functionality with rules, and they all were able to create

Related Work in Context-Based Customization

Bill Schilit, Norman Adams, and Roy Want used the location context of wireless active badge devices to activate reminders or to show the nearest printer.[1] The environment's infrastructure triggered these actions and tracked the location of the badges, which users carried. Peter Brown, John Bovey, and Xian Chen introduced Stick-e notes—virtual reminder tags activated by context information, such as location, people, or a share price.[2] Developers configured the activation conditions for displaying the note and its text using a textual, SGML-based language. Albrecht Schmidt and his colleagues presented a prototype system for abstracting contexts using sensors for acceleration, touch, light, and so on.[3] Their prototype used the abstracted contexts to adjust a phone's profile.

This early research concentrated on specific applications; it didn't address the general problem of context-based end-user customization for mobile devices, where a scalable, easy-to-use GUI can connect any event or state to any action.

Recent research focuses on customizing using a large screen (such as a laptop PC) in a smart environment. Timothy Sohn and Anind Dey discuss an informal pen-based prototype tool for configuring, in a PC environment, input devices that collect context information and output devices that support response.[4] Developers can combine inputs and outputs into rules and test them with the tool. As a future research topic, Sohn and Dey identify the need to give both designers and end users the ability to create and modify context-aware applications. In other work, Dey and his colleagues present a programming-by-demonstration approach for prototyping context-aware applications.[5] They experimented with a prototype PC-based tool that lets users train and label context models, which can then be mapped to actions. The researchers defined context models as examples.

Khai Truong, Elaine Huang, and Gregory Abowd introduce an end-user application-programming approach involving automated capture and playback of home activities.[6] Their system is based on a magnetic-poetry metaphor. The system's PC-based user interface contains a set of predefined, domain-specific words that users can arrange to define system functionality. To function, each user-defined application must include context information for time, duration, frequency, location, and people.

Unlike other research, our approach is mobile-device-centric, not infrastructure-centric. Our software framework, customization tool, and sensors are located in a smart phone that tracks the environment, rather than having the environment track the smart phone. This approach lets users customize the mobile device and use it anywhere, independent of external infrastructure. Finally, unlike existing research, our approach offers real-time pattern recognition of freely user-trainable acceleration-based gestures in a mobile device. It also features an RFID tag read/write accessory. Both modalities are customizable to activate any available mobile-device functions.

REFERENCES

1. B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications," *Proc. 1st Int'l Workshop Mobile Computing Systems and Applications*, IEEE CS Press, 1994, pp. 85–90.
2. P. Brown, J. Bovey, and X. Chen, "Context-Aware Applications: From the Laboratory to the Marketplace," *IEEE Personal Comm.*, vol. 4, no. 5, 1997, pp. 58–64.
3. A. Schmidt et al., "Advanced Interaction in Context," *Proc. 1st Int'l Symp. Handheld and Ubiquitous Computing*, LNCS 1717, Springer, 1999, pp. 89–101.
4. T. Sohn and A. Dey, "iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications," extended abstracts, *Proc. Int'l Conf. Computer-Human Interaction* (CHI 04), ACM Press, 2003, pp. 974–975.
5. A. Dey et al., "a CAPpella: Programming by Demonstration of Context-Aware Applications," *Proc. Int'l Conf. Computer-Human Interaction* (CHI 04), ACM Press, 2004, pp. 33–40.
6. K. Truong, E. Huang, and G. Abowd, "CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home," *Proc. Int'l Conf. Ubiquitous Comp.* (Ubicomp 04), 2004, Springer, 2004, pp. 143–160.

the correct scenario functionality. Our success here hinged on an earlier observation: users prefer very simple rules in customization.[3] We therefore defined a minimal rule structure, incorporating only the logical operator AND. When users start making rules, they assume AND as the default operator, even though it isn't explicitly shown. The logical operator OR is available by creating parallel rules.

However, even simple rules don't guarantee complete user control if the user is totally unfamiliar with a certain trigger's function. The tool should therefore provide a more detailed explanation about each trigger's exact meaning—as a Help function, for example, or a visualization. Furthermore, users might want to define and name their own context triggers, such as locations or social situations based on Bluetooth devices, temperature abstractions, light levels, and so on. We plan to provide a GUI for personalizing specific context sources and letting users update the vocabularies at runtime. In the current prototype, users can train and name the gesture-type triggers, and name RFID tags by physically writing tags with the RFID tag read/write accessory. They can also update the vocabularies using a simple text editor.

Cost of learning

In end-user development,[1] the general goal is to achieve low learning cost while still offering users a range of customizable functionality. Our evalu-

ation confirmed the tool's low learning cost. As for functionality scope, our rule-based customization approach applies best to actions that operate using a single discrete command. We demonstrated several such applications, and users had dozens of actions available for customization. However, our approach isn't well suited to customization tasks involving continuous control tasks (such as moving a cursor) or extensive keyboard use (such as text input). Users can realize task sequences by creating multiple rules for the same trigger. Ultimately, however, we had to trade off customization scope to achieve high tool usability.

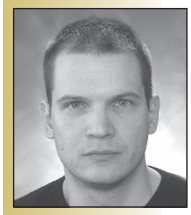
Because developers can use the context framework and our customization tool as a general platform for connecting any abstracted, sensor-based event to application control, the system facilitates rapid development and evaluation of new sensor-based input modalities. Such emerging modalities, which users can customize to their preferences, potentially increase the efficiency of human-computer interaction with smart phones. As interaction convergence on a single device increases, so too will the number of different external devices that a smart phone can control.

Our future work will include developing new input and output modalities for smart phone interaction and evaluating them through realistic use cases. Also, once more hardware copies are available, we plan to test the prototype's use in people's daily lives. P

ACKNOWLEDGMENTS

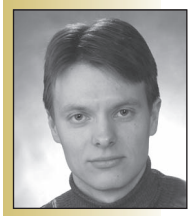
Our work is the result of successful system integration and cooperation. Antti Takaluoma and Heikki Huomo played a significant role in developing the RFID tag accessory. Sanna Kallio, Heikki

the AUTHORS



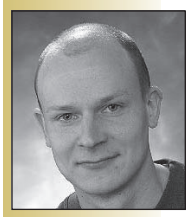
Panu Korpipää is a senior research scientist at the Technical Research Centre of Finland, VTT Telecommunications, in the Mobile Interaction Center. His research interests include multimodal, sensor-based, and situational interaction in mobile computing, and end-user development. He received his PhD in information processing from the University of Oulu. Contact him at VTT Telecommunications, PO Box 1100, FIN-90571 Oulu, Finland; panu.korpipaa@vtt.fi.

Jonna Häkkilä is a usability expert at Nokia Multimedia Research and Development in Oulu, Finland. Her research interests include context awareness and human-computer interaction, including mobile, ubiquitous, and multimodal user interfaces. She's finalizing her PhD in information processing from the University of Oulu. Contact her at Nokia Multimedia, Yrttipellontie 6, FIN-90230 Oulu, Finland; jonna.hakkila@nokia.com.



Esko-Juhani Malm is a research scientist at the Technical Research Centre of Finland, VTT Telecommunications, in the Mobile Interaction Center. His professional interests include embedded software engineering and pervasive computing. He received his MSc in information processing from the University of Oulu. Contact him at VTT Telecommunications, PO Box 1100, FIN-90571 Oulu, Finland; esko-juhani.malm@vtt.fi.

Tapani Rantakokko is a research scientist at the Technical Research Centre of Finland, VTT Telecommunications, in the Mobile Interaction Center, and a doctoral student in the University of Oulu's Department of Electrical and Information Engineering. His research interests include mobile user interfaces and pervasive computing. He received his MSc in information processing from the University of Oulu. Contact him at VTT Telecommunications, PO Box 1100, FIN-90571 Oulu, Finland; tapani.rantakokko@vtt.fi.



Vesa Kyllönen researches advanced interactive systems at the Technical Research Centre of Finland, VTT Telecommunications. His research interests include evolutionary computing and software engineering. He received his MSc in information processing from the University of Oulu. Contact him at VTT Telecommunications, PO Box 1100, FIN-90571 Oulu, Finland; vesa.kyllonen@vtt.fi.

Juha Kela is a research scientist at the Technical Research Centre of Finland, VTT Telecommunications, in the Mobile Interaction Center, and a doctoral student in the University of Oulu's Department of Electrical and Information Engineering. His research interests include multimodal and mobile interaction. He has an MSc in information processing from the University of Oulu. Contact him at VTT Telecommunications, PO Box 1100, FIN-90571 Oulu, Finland; juha.kela@vtt.fi.

Ilkka Känsälä is an R&D engineer at Avantone and a doctoral student in the University of Oulu's Department of Electrical and Information Engineering. His research interests include ubiquitous, pervasive, and context-aware computing; biometrics; and technologies for adaptive user interaction. He received his MSc in information processing from the University of Oulu. Contact him at Avantone Inc., Kasarmintie 15, FIN-90100 Oulu, Finland; ilkka.kansala@avantone.com.

Jani Mäntyjärvi is a senior research scientist at the Technical Research Centre of Finland, VTT Telecommunications, in the Mobile Interaction Center. His research interests include pervasive and context-aware computing for handheld devices and technologies for adaptive user interaction. He received his PhD in information processing from the University of Oulu. Contact him at VTT Telecommunications, PO Box 1100, FIN-90571 Oulu, Finland; jani.mantyjarvi@vtt.fi.

Keränen, Jari Kangas, and Samuli Silanto helped develop the gesture modality. Harri Lakkala and Ilkka Salminen helped develop context sources and abstractors and provided the rule script engine. Sami Ronkainen and Kirsi-Maria Hiltunen contributed to the user interface design and to organizing the user tests. Urpo Tuomela has provided valuable guidance. Finally, we thank our user test participants and the people who evaluated the vocabularies.

REFERENCES

1. G. Fischer et al., "Meta-Design: A Manifesto for End User Development," *Comm. ACM*, vol. 47, no. 9, 2004, pp. 33–37.
2. J. Mäntyjärvi et al., "Context-Studio–Tool for Personalizing Context-Aware Applications in Mobile Terminals," *Proc. Australasian Computer Human Interaction Conf.*, Addison-Wesley Longman, 2003, pp. 64–73.
3. P. Korpipää et al., "Utilising Context Ontology in Mobile Device Application Personalisation," *Proc. Int'l Conf. Mobile and Ubiquitous Multimedia*, ACM Press, 2004, pp. 133–140.
4. P. Korpipää et al., "Managing Context Information in Mobile Devices," *IEEE Pervasive Computing*, vol. 2, no. 3, 2003, pp. 42–51.
5. H. Lakkala, *Context Exchange Protocol Specification*, Nokia Corp., 2003; www.mupe.net/files/cep_1_0.pdf.
6. H. Lakkala, *Context Script Specification*, Nokia Corp., 2003; www.mupe.net/files/context_script_1_0.pdf.
7. J. Kela et al., "Accelerometer-Based Gesture Control for a Design Environment," to be published in *Personal and Ubiquitous Computing*, Springer; available online at www.springerlink.com/index/N813460153P65L5M.pdf.
8. H. Ailisto et al., "A Physical Selection Paradigm for Ubiquitous Computing," *Proc. 1st European Symp. Ambient Intelligence* (EUSAI 03), LNCS 2875, Springer, 2003, pp. 372–383.
9. J. Nielsen and R.L. Mack, *Usability Inspection Methods*, John Wiley & Sons, 1994.
10. A. Dey et al., "a CAPpella: Programming by Demonstration of Context-Aware Applications," *Proc. Int'l Conf. Human Factors in Computing Systems* (CHI 04), ACM Press, 2004, pp. 33–40.
11. K. Truong, E. Huang, and G. Abowd, "CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home," *Proc. Int'l Conf. Ubiquitous Computing* (Ubicomp 04), Springer, 2004, pp. 143–160.