Ubi-Designer: A Web-Based Toolkit for Configuring and Field-Testing UbiComp Prototypes

Martijn H. Vastenburg
Faculty of Industrial Design Engineering
Delft University of Technology
Landbergstraat 15
2628 CE, Delft, The Netherlands
+31-15-2784960
M.H.Vastenburg@tudelft.nl

Halldór Fjalldal, Charles van der Mast
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD, Delft, The Netherlands
+31-15-2782549
H.Fjalldal@student.tudelft.nl
C.A.P.G.vanderMast@tudelft.nl

ABSTRACT
Technology is now available for creating affordable sensor networks and infrastructures for ubiquitous computing environments. In the area of ambient assisted living, context-awareness is considered to be a key factor towards creating acceptable solutions that support elderly people in living independently in their homes as long as possible. Unfortunately, at the present state of technology, the design of context-aware products and services requires substantial technical knowledge. Consequently, product designers are often dependent on engineers for implementing prototypes and consequently prototyping their design concepts is a costly and time-consuming process. This paper presents a web-based toolkit that supports product designers in prototyping and configuring interactive context-aware services in multiple homes. The toolkit has been designed and tested in close collaboration with interaction designers. Using the toolkit, designers can make fast design iterations and eventually lower development cost.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *prototyping, user-centered design.*

General Terms
Design, Algorithms.

Keywords
Ubiquitous computing, context-aware products and services, design, toolkits, prototyping.

1. INTRODUCTION
The vision of ubiquitous computing is getting closer to reality. Technology is in place for creating environments that sense the state of the environment and the activities of its inhabitants; services can be developed that pro-actively adapt the environment to the actual user needs. Examples of state-of-the-art systems include applications for monitoring and stimulating social connectedness [9], physical play [11], and stimulating physical exercise [10].

Even though hardware components needed for building context-aware products and services are generally available, it tends to be hard to link the components together. Data from the sensors need to be collected, stored, and interpreted, and the system needs to react accordingly. In case of a standalone product with only few sensors attached, implementation of a design concept can be simple. In case of a distributed system, e.g., several houses with participants, a range of sensors and complex functionality, the development and deployment of a prototype can be troublesome.

This paper describes a toolkit for supporting designers of interactive context-aware products and services. First, the existing product design process is studied in relation to context-aware products and services. Second, the requirements for a toolkit supporting the designers are identified. Third, the design of the toolkit is described. Fourth, the results of the evaluation of the toolkit with a panel of industrial designers are presented.

2. DESIGN PROCESS
In order to better understand the needs of the designers, we started with studying the current design practice. The design process generally consists of several stages. Based on a series of observations of product design processes, Buxton identified three design stages as depicted in Figure 1 [2]. First, the *design* stage represents a creative phase in which product ideas are explored and the feel and interaction involved in performing the desired functions are studied. Many designers start with design sketches; these sketches evolve into working prototypes that capture the essence of the designed concepts. These prototypes enable people to experience the feel and interaction. The design stage is concluded by a clear description of the projected product.

Figure 1. The design process of products can be broken into three stages: a creative stage ("design"), a development stage ("engineering"), and a marketing and sales stage ("sales").

PETRA'09, June 09-13, 2009, Corfu, Greece.

Whereas the design stage tends to be creative and explorative, the *engineering* stage can be characterized as a goal-oriented implementation phase. Even though the technical details yet need to be filled in, the requirements describing the projected product unambiguously define the desired outcome. Most of the time, the prototypes used in the design phase will not be re-used in the engineering phase; the engineering goals usually require rebuilding from scratch.

The *sales stage* concludes the product design process. After the engineering stage has finished, the product is ready for shipping.

Buxton emphasizes the differences between the design stage and the engineering stage. The design stage is a creative stage that requires creative people. In his observations, designers tend to be creative and innovative, whereas engineers tend to be extremely well organized and technically strong. One needs both designers and engineers to complete the design process successfully. Designers and engineers have a different job to do, and likewise they need different tools to do the job.

Both designers and engineers advocate the notion of rapid prototyping and iterative design, but the purpose of developing prototypes differs. In the design stage, prototypes are used to experience the feel and interaction; the technology used is of minor relevance. In the engineering stage, on the other hand, the focus is on technology; prototypes are used to test technical aspects of the hardware and software design.

Dow et al. (2006) studied the design stage in order to be able to define the appropriate tools for professional designers. Rather than taking a technology-centered approach, they took a design-centered stance. The central challenge underlying their study was to find out how professional designers externalize ideas for off-the-desktop computing, and how these ideas inform next generation design tools. Observations and interviews with 11 designers revealed that whereas the design process differs from one designer to another, the general flow underlying the process is the same. Figure 2 shows the design steps underlying the creative design stage, as defined by Dow et al.

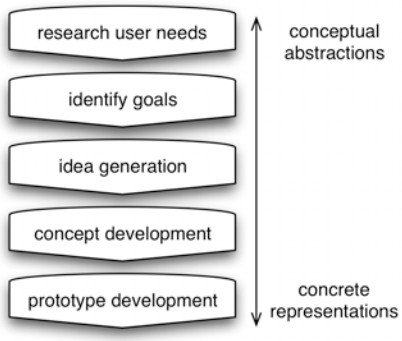

Figure 2. The design steps underlying the creative design stage (adapted from [4]).

The first steps of the design stage tend to be abstract. Designers define the problem domain and explore the design space using e.g., focus groups, probes and brainstorm sessions. Early design ideas are generally paper-based, for example using story boarding or scenarios. The design ideas evolve into working prototypes in which the look and feel of the design concept can be experienced by both the designers and by the target users. The result of the design stage is a concrete description of a product concept that is ready to be constructed by engineers. This product design process has been used as a reference in the development of the Ubi-Designer toolkit as described in the following sections.

3. RELATED WORK

As discussed in the previous section, the product design process involves not only designers, but also engineers. Most of the existing toolkits for developing context-aware products are focused on engineers, for example the Context Toolkit [3] and the Pervasa Atlas platform [8]. These toolkits enable engineers to configure sensors, abstract sensor data, and feed services with abstracted information. The toolkits generally give access to all details of the systems, thereby providing high flexibility, at the cost of learnability. Whereas these toolkits are flexible and powerful, they tend to be hard to use for non-engineers.

Several toolkits have been developed targeted at designers. These toolkits generally support the iterative development of prototypes in an easy-to-understand way, rather than focusing on scalability and low footprint in terms of memory usage and processing power. Notably, many industrial designers and design students use MaxMSP[1], a visual programming environment, for building context-aware prototypes. MaxMSP is a dataflow-oriented programming environment, in which for example sensors can be visually linked to processors and actuators. In our observations, MaxMSP was found to work best for standalone prototypes that are based on standard library components. When a certain prototype requires non-standard components, implementation can be hard. Furthermore, management of a distributed set of sensors using MaxMSP is complex; scalability certainly is an issue here.

Whereas MaxMSP is targeted at sensor-based applications in general, CollaborationBus is targeted specifically at ubiquitous computing applications [6]. Users can define system behavior by specifying the information flow from sensors to actuators. The status of sensors and information flow is visualized in the application in real-time. The application provides an elegant solution for configuring and monitoring a ubiquitous system in a single home. Since CollaborationBus does not provide a mechanism to cluster sensors and information flows, it is currently not usable for a prototype test in multiple homes using a single server. Furthermore, CollaborationBus lacks easy access to the system via a web interface.

Next to the tools for professionals, there are also tools targeted at non-expert users. For example, the Jigsaw editor enables non-expert users to visually build a ubiquitous environment [7]. Devices are represented as jigsaw pieces that can be dragged and assemble to build a ubiquitous environment. Similarly, iCap allows end-users to visually program their system by defining situations and actions without writing any code [3]. Whereas these tools make it easy to construct simple prototypes, it is hard if not impossible to create a more complex system. Furthermore, none of the end-user-programming tools is capable of managing multiple homes.

[1] www.cycling74.com/products/maxmsp

4. PROBLEM STATEMENT

The toolkits as discussed in section 3 show how context-aware systems can be configured and managed by end-users, designers and engineers. Whereas toolkits are available for single-house applications, there are no tools available that support designers in creating applications for a series of houses. In the current trend of living labs and longitudinal testing in the field, there is an actual need for a design tool that does cover multiple houses.

The goal of this project is to provide designers with a tool that they can use to easily prototype design concepts, and test these prototypes in multiple homes in the field. The tool should enable designers to link distributed sensor nodes to a central system, and to link distributed services to the same central system. The focus will be on the domestic application domain.

5. REQUIREMENTS

Based on observations of design practices, discussions and interviews with industrial designers at the Faculty of Industrial Design Engineering in Delft, the overall design goal was translated into the following list of high-level user requirements.

REQ1. Enable designers to configure the system.
Whereas a programmer might be needed to initially setup a prototype system, designers indicated that they prefer to be able to configure and change the system themselves.

REQ2. Enable designers to easily monitor the status of sensors and services.
Designers indicated that it can be hard to monitor a prototype when it is being tested in the field. It should be possible to check the status of the system remotely.

REQ3. Provide flexible mechanism for defining situations.
Designers indicated that they want to be able to define situations in relation to sensor values (e.g., temperature=20 ^{o}C), abstracted information (e.g., user activity="cooking"), statistics (e.g., average temperature=20 ^{o}C), and temporal patterns (e.g., # toilet visits in the morning).

REQ4. Enable designers to link actions to situations.
Designers indicated that they prefer to be able to define system behavior themselves. System behavior is generally defined by linking actions to situations; when a situation is recognized by the system, the appropriate action is activated. In practice, the situations and actions often need to be adapted to specific settings.

REQ5. Enable designers to fake sensors and simulate events.
Designers indicated that they would like to test a prototype system using both real data and simulated events.

REQ6. Support a multi-home setting.
Increasingly, prototypes for pervasive health applications are tested in multiple houses at the same time. The inherent complexity in sensors and infrastructure results in problems when configuring and monitoring the system. A toolkit should provide a way to manage the complexity of a multi-home test setting.

REQ7. Make platform open to external hardware and software components.
Sensors, communication hardware, and data processing software differ from project to project. A toolkit should be open to new components.

6. UBI-DESIGNER TOOLKIT

In developing the toolkit, the primary aim was to create a setting in which designers could develop, test and improve their prototypes in quick iteration cycles. The toolkit would need to give access to the required functionality without burdening the designers with technical details. Ideally, a designer could build and test a context-aware prototype from scratch without support from an engineer.

There were however two obstacles that could not be solved without support of engineers:

1) *Linking new sensors to the system*. There is no standard set of sensors available. Designers tend to explore the design space, and therefore they often use non-standard sensors. Since there are no industry-wide protocols for communication with sensors, engineers are needed to interface new sensors with a context-aware system.

2) *Writing non-trivial algorithms for interpreting sensor data.* Most designers are able to analyze data using basic functions. In many situations, however, data analysis requires a skilled programmer.

Whereas a generic context-aware system, consisting of a database, a reasoner and communication infrastructure, can well be used as a start, an engineer is needed to set up project-specific sensor configuration and data processors. With all infrastructural components in place, however, a designer could well monitor and adapt the system without help of an engineer.

To achieve this separation between low-level technical access and high-level 'creative' access to the system, it was decided to create a layered system consisting of a *platform* that entails the low-level middleware functionality needed to deploy a context-aware prototype, and a web-based *toolkit* that provides high-level access to the platform (figure 4). The platform collects the data from the sensors in the field, processes the data, and sends events to registered services. For this project, a basic generic platform was

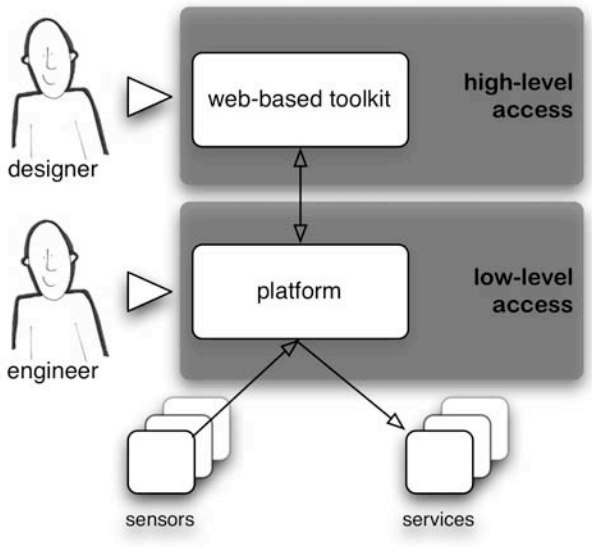

Figure 4. The system design consists of two layers: the platform layer provides low-level access to engineers, and the toolkit layer provides high-level access to designers.

implemented. The web-based toolkit can be used to configure and monitor the sensors, data interpretation mechanisms and the events. Using the web interface, designers can monitor and update the configuration both in the field (when setting up the prototype) and from a remote location (when monitoring a field test).

The UBI-Designer toolkit enables designers to change the configuration of the sensors, the software algorithms for processing sensor data, and the rules that trigger the actuators.

6.1 Sensors

The toolkit shows a list of all sensors attached to the system, including the location and the status of the sensors (Figure 5). The designer can switch between a general overview and a filtered view showing the sensors for a specific project (e.g., house) only. Sensor values can be simulated (by selecting the sensor in the list, and pressing "simulate"), which makes it easier for designers to test the system behavior.

New sensors are automatically listed when they are registered with the context-aware service platform. Sensors register to the system using a simple protocol, which has to be implemented for new sensor types by an engineer. Likewise, a sensor can be simulated, by linking a piece of software to the platform that mimics the behavior of a real sensor. Using these *virtual sensors*, a system can be tested without all hardware in place.

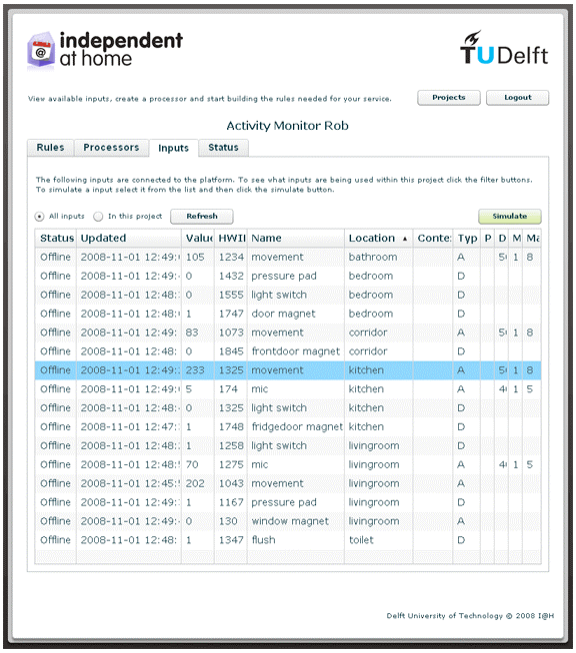

Figure 5. The input view shows the status of all sensors.

All sensors communicate to the central platform through the Internet. A message-based protocol is used to communicate the sensor status and sensor values to the platform. The Internet-based communication makes it easy for designers to deploy sensors in multiple locations in the field.

6.2 Processors

Low-level interpretation of sensor data, using for example pattern recognition, is facilitated using *processors*. These processors are virtual sensors that can are treated the same way as sensors. A library of processors is readily available to be used by the designers. Since creation of new processors requires extensive programming skills, it was decided to shield the details of processors from the designers. Engineers can create new processors in Java using a template, and add the processors to the library. Figure 6 shows the panel that enables designers to activate and configure the processors from the library. Processor values can be simulated similar to sensor values; this way, designers can easily test the link between processors and services.

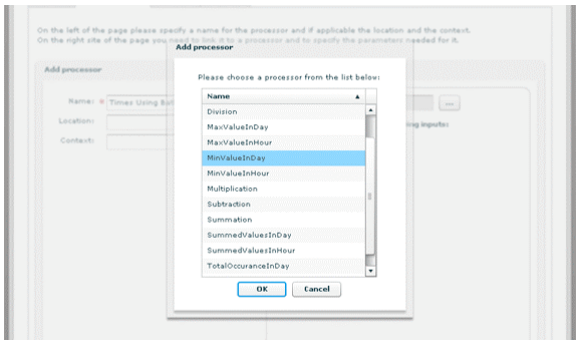

Figure 6. An overview is given of all processors available for interpreting low-level sensor data.

6.3 Rules

Designers can define situations and actions using a rule-editor. Each rule links a situation to an event; these events trigger services that are linked to the platform. For example, a high-temperature situation could be linked to an event, which automatically activates the air-conditioning. It was decided to provide simplified access to the JESS rule engine [5]; the JESS engine itself was integrated in the platform, hidden from the designer's eye. The Ubi-Designer rule-editor (Figure 7) provides simplified access to the JESS rule engine. To speed up the rule-definition, a pre-defined list of templates is available that covers often-used constructs.

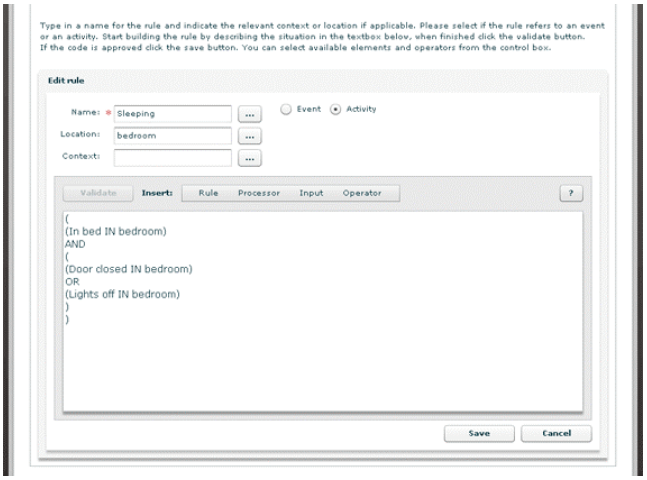

Figure 7. New rules can be defined using a rule-editor, which provides simplified access to a rule engine.

The *rules*-panel (Figure 8) can be used to simulate events. Each rule can be simulated, which makes it easier for designers to test the system behavior. The *active* attribute indicates the status of the rule; a rule is active whenever the situation as defined by the conditional elements is being matched.

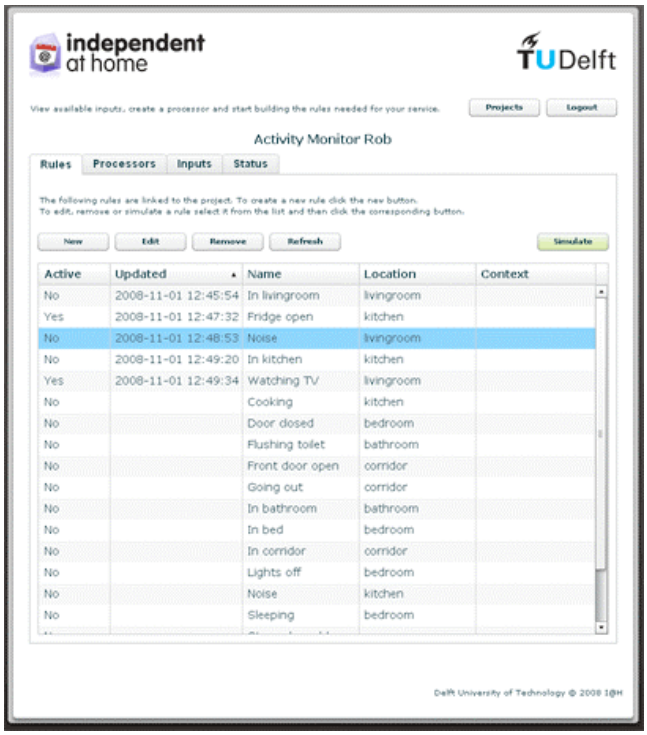

Figure 8. Using the rules view, designers can check the status of the rules, and simulate events.

6.4 Services

The context-aware platform is primarily used to provide services with relevant context information. For example, a medicine reminder service needs to know when medicine is taken. For most of the design projects observed, the services were developed in Flash. The platform will collect sensor data and recognize those situations that are related to medicine intake using service-specific rules. The service is notified of these situations using XML messages.

When using Ubi-Designer, services need to register themselves at the platform by sending a predefined XML message to the server. As part of the registration process, services have to register for events. The *status panel* (Figure 9) shows the status of all services connected to the platform.

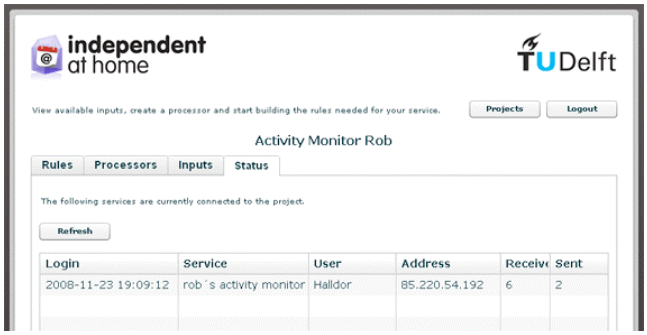

Figure 9. The status pane provides an overview of the status of the services that are connected to the platform.

6.5 Projects

When talking to industrial designers, it became clear that the number of sensors, processors and rules can be high. For a single house, the use of 30 sensors is not uncommon. Whereas managing a context-aware prototype in a single house can be hard, it is even more complex to manage a context-aware prototype in a multi-home setting. A multi-home setting requires not only a higher number of sensors, but also a higher number of processors, rules and services, often with different settings for each participant. Therefore it was decided to add a clustering mechanism to the toolkit. Each sensor, processor, rule and service can be linked to a *project*. A designer can select a project in the toolkit, thereby focusing only on relevant components. Using projects, the complexity of a multi-home system has been reduced.

7. Evaluation

The Ubi-Designer toolkit has been evaluated with designers in a formative evaluation. The evaluation aimed to find out if the toolkit supports designers in easily prototyping design concepts in a multi-home setting. Three service designers were asked to use the toolkit for configuring and prototyping an imaginary remote monitoring-service following a predefined scenario. The participants were experienced in interaction design, and they were all currently working on designing context-aware applications. The participants used their own personal computer to access the web-based interface of the toolkit. A paper-based tutorial describing the steps that had to be taken was given to the designers. In the assignment, the participants had to create new rules to detect situations, and they had to link a service application to the platform. After finishing the assignment, participants were asked to rate the overall experience using a questionnaire, which was based on the usability scales as proposed by Benyon et al. (2005). The questionnaire concluded with seven questions in which the participants were asked for their experiences and for feedback.

The participants took about 30 minutes to complete the assignment; the questionnaire took about 15 minutes per participant. The average user ratings are shown in Table 1. According to the user ratings, the toolkit was considered very useful for prototyping context-aware services. All three participants rated the toolkit the maximum score on *usefulness when prototyping activity-aware services.* The average scores on *ease-of use, visual design quality, trustworthiness* and *pleasantness in use* all range between +1 and +2.

Table 1. Averaged user ratings on the toolkit user interface (n=3).

Item	Average score (-2=low, +2=high)
Ease-of-use	+1
Usefulness	+2
Visual design quality	+1.33
Trustworthiness of the toolkit	+1.66
Pleasantness in use	+1.33

In general, the participants could easily configure the assigned service. They did however need some time to get familiar with the rule syntax. They suggested adding a tutorial that would explain how rules can be defined. Furthermore, the participants would like to have a rule editor that allowed more expressive constructs; apparently, the rule editor in the prototype was experienced as too simplified. Next to these general comments, there were some detailed remarks that help streamlining the interface; for example, rather than manually validating new rules, the participants would like the toolkit to automatically validate rules.

Based on the results of the formative evaluation, the rule editor will be improved. As a next step, the toolkit will be evaluated by a panel of approximately 10 designers who will be using the toolkit in a realistic setting with real design cases.

8. DISCUSSION AND FUTURE WORK

This paper proposed a web-based toolkit that can be used by designers for configuring and monitoring context-aware applications. Using the toolkit, designers can easily configure and monitor context-aware systems, whereas traditional toolkits would require a skilled engineer. The toolkit enables designers to make quick design cycles. This makes it easier to involve end users in the design process, since the feedback of users can now be incorporated in the prototype without getting back to engineers. Compared to traditional toolkits for creating context-aware systems, the Ubi-Designer toolkit combines the flexibility and ease-of-use of design-oriented solutions with scalability and robustness of technology-oriented solutions.

The toolkit has been developed in close collaboration with designers, and a formative evaluation has been conducted with three designers. As a next step, the toolkit will be applied in a series of design cases, in order to collect feedback in a realistic setting. More information on the toolkit and the second stage of the evaluation can be found online[2].

In terms of shielding complexity from designers, we do need to find the right balance between expressiveness and ease-of-use. Whereas many of the technical details can easily be shielded from the designers, the designers would like to have full control in configuring the rules. A new version of the toolkit will therefore include an improved rule-editor that offers full control when defining rules.

9. ACKNOWLEDGMENTS

The work presented in this paper was part of the Independent at Home project (MMI06011), funded by SenterNovem through the IOP-MMI program. The authors would also like to thank the industrial designers who participated in the design and evaluation of the toolkit.

[2] http://independentathome.tudelft.nl/toolkit

10. REFERENCES

[1] Benyon, D., Turner, P., and Turner, S. 2005. Designing Interactive Systems: People, Activities, Contexts, Technologies. Addison-Wesley.

[2] Buxton, W. 2003. Performance by Design: The Role of Design in Software Product Development. In Proceedings of the Second International Conference on Usage-Centered Design, 1-15.

[3] Dey, A. K., Sohn, T., Streng, S., and Kodama, J. 2006. iCAP: Interactive Prototyping of Context-Aware Applications. Pervasive 2006, Springer LNCS 3968, 254-271.

[4] Dow, S., Saponas, T. S., Li, Y., and Landay, J. A. 2006. External Representations in Ubiquitous Computing Design and the Implications for Design Tools. In Proceedings of the 6th Conference on Designing Interactive Systems, 241-250.

[5] Friedman-Hill, E. 2003. JESS in Action: Rule-Based Systems in Java. Manning Publications Co.

[6] Gross, T., and Marquardt, N. 2007. CollaborationBus: An Editor for the Easy Configuration of Ubiquitous Computing Environments. In Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 307-314.

[7] Humble, J., Crabtree, A., Hemmings, T., Åkesson, K.-P., Koleva, B., Rodden, T., and Hansson, P. 2003. "Playing with the Bits" User-Configuration of Ubiquitous Domestic Environments. In Proceeding of the 5th Conference on Ubiquitous Computing, Springer LNCS 2864, 256-263.

[8] King, J., Bose, R., Hen-I Yang, Pickles, S., and Helal, A. 2006. Atlas: A Service-Oriented Sensor Platform: Hardware and Middleware to Enable Programmable Pervasive Spaces. In Proceedings of the 31st IEEE Conference on Local Computer Networks, 630-638.

[9] Morris, M. E. 2005. Social Networks as Health Feedback Displays. IEEE Internet Computing 9(5), 29-37.

[10] Vastenburg, M. H., Visser, T., Vermaas, M., and Keyson, D. V. 2008. Designing Acceptable Assisted Living Services for Elderly Users. European Conference on Ambient Intelligence. Springer LNCS 5355, 1-12.

[11] Wakkary, R., Hatala, M., Lovell, R., and Droumeva, M. 2005. An Ambient Intelligence Platform for Physical Play. In MULTIMEDIA '05: Proceedings of the 13th Annual ACM International Conference on Multimedia, 764-773.