Utilising Context Ontology in Mobile Device Application Personalisation

Panu Korpipää[1], Jonna Häkkilä[2], Juha Kela[1], Sami Ronkainen[2], Ilkka Känsälä[2]
[1)]VTT Electronics
P.O.Box 1100, FIN 90571 Oulu, Finland
Email: firstname.lastname@vtt.fi
[2)]Nokia
P.O.Box 300, FIN 90401, Finland
Email: firstname.lastname@nokia.com

ABSTRACT
Context Studio, an application personalisation tool for semi-automated context-based adaptation, has been proposed to provide a flexible means of implementing context-aware features. In this paper, Context Studio is further developed for the end users of small-screen mobile devices. Navigating and information presentation are designed for small screens, especially for the Series 60 mobile phone user interface. Context ontology, with an enhanced vocabulary model, is utilized to offer scalable representation and easy navigation of context and action information in the UI. The ontology vocabulary hierarchy is transformed into a folder-file model representation in the graphical user interface. UI elements can be directly updated, according to the extensions and modifications to ontology vocabularies, automatically in an online system. A rule model is utilized to allow systematic management and presentation of context-action rules in the user interface. The chosen ontology-based UI model is evaluated with a usability study.

Categories and Subject Descriptors
H.5 [**Information Interfaces and Presentation**]: User Interfaces – evaluation/methodology, *graphical user interfaces*, *user interface management systems*

General Terms
Design

Keywords
Context awareness, ontology, mobile device, application personalization, user interface, rule, Context Studio

MUM'04, October 27–29, 2004, College Park, Maryland, USA.

1. INTRODUCTION
The notion of adapting application behavior to context has received critique in the literature. Even though the goal of context-aware computing is well founded, developing devices that can sense the situation and adapt their actions appropriately faces one foundational problem; context awareness exhibited by people is radically different from that of computational systems [4]. Hard-coded fully automatic actions based on context are rarely useful, and incorrect automatic actions can be frustrating. Greenberg points out that it is not always possible to enumerate a priori a limited set of contexts that match the real-world context [6]. If such a set is found and is valid today, it may be inappropriate at any other time because of "internal and external changes in the social and physical circumstances". Moreover, determining an appropriate action from a given context may be difficult.

However, it is not necessary to aim at fully automated actions as the only goal of context awareness. Mäntyjärvi et al. propose to use an automation systems categorization for context-based adaptation [11]. Three different levels of automation of context-dependent actions can be distinguished: manual, semi-automated, and fully automated. Manual actions refer to actions made by the user based on context information, which is detected by the device (or the user). At the semi-automated level the user may predefine application actions based on context detected by the device, or choose from the actions proposed by the device based on context. At the fully automated level the application automatically takes (pre-programmed) actions according to the context detected by the device.

The semi-automated adaptation model partially overcomes the problem [6] of determining an appropriate action based on context. If the event-action behavior is defined by the end user instead of the application developer, a greater degree of personalization and flexibility can be achieved. Further flexibility is achieved by letting the user change the event-action configurations if it is required when the circumstances change over time.

Ranganathan and Campbell introduce a first-order logic-based rule model and distributed framework for defining context-based application actions [13]. As further work, the authors identify developing a graphical interface that lets the user choose the

elements of a rule from the available contexts and actions, instead of writing first-order logic. Sohn and Dey introduce an informal pen-based tool that allows users to define input devices that collect context information, and output devices that support response [15]. Inputs and outputs can be combined into rules and tested with the tool. The tool is not designed for small-screen mobile devices, nor are the user interface elements generated based on an explicit information model. The authors identify as future work the goal of enabling both designers and end users to create and modify context-aware applications. Dey et al. experiment with a programming-by-demonstration approach for prototyping context-aware applications [2]. The authors have developed a tool that allows the user to train and annotate models of context by example, which can be bound to actions. Again, the user interface of the tool is not designed for small-screen mobile devices. Furthermore, the focus differs from that chosen in this paper, which discusses a scalable model for representing the contexts of an ontology in the UI, where the contexts are freely selectable by the user.

The approach of learning contexts by example from a set of primitives from multisensor data has a fundamental problem: all primitives function as inputs for every context, acceleration, orientation, temperature, sound, light, humidity, heart-beat, etc. Therefore, when the user attempts to train the device in a certain context, the training data usually contains irrelevant inputs that are unintentionally included in the model. For example, if the user trains walking by walking near a road with a high rate of traffic, the model may learn that traffic noise is a part of walking and not recognize it without it. On the other hand, if walking is trained in silence, it may not be recognized in noise. With an accurately predefined ontology for predefined sensor and other inputs, and accurate rule definition by the user, context-action rules will only contain those inputs that the user wants to have.

Context Studio is an application personalization tool for semi-automated context-based adaptation. Application personalization is performed with a graphical user interface that allows the user to bind contexts to application actions. The Context Studio concept was introduced by Mäntyjärvi et al., who also presented a usability evaluation for a proof-of-concept prototype of the tool [11]. This paper utilizes the results and experiences from the study and further develops the concept; the usability problems found in the earlier study are corrected, and a user interface designed for small screens, especially Series 60 mobile devices, is evaluated with users novel to the concept of context-awareness. Furthermore, context and action vocabulary models, a rule model, and a context framework [8] are utilized in the new prototype. These models enable dynamic management of rules and extensible context vocabularies, and enable automatically generating elements of the user interface at runtime.

Context ontology can be applied for enumerating all the possible context events that can be used for activating actions. Many definitions for ontologies are available [5]; the purpose of ontology in this paper is expressing information so that it is easily understandable by humans and readable by machines. The human understandability of context information enables the mobile device end user to configure context-aware features. The machine readability of context information enables dynamically forming user interface elements based on the ontology. Moreover, the ontology facilitates describing rules as Context Exchange Protocol (CEP) XML scripts, which can be executed by an inference engine [10]. Many context models and ontologies have appeared in the literature, e.g. [3,14,16]. The ontology model for a sensor-based mobile context [7,8] is utilized in this study. A more detailed vocabulary model is presented as an enhancement to the previous model.

Ontologies can be divided into lightweight and heavyweight, based on the degree of “depth” and restrictions (axioms, constraints) on domain semantics. The ontology applied in this paper is considered a lightweight ontology, including concepts, concept taxonomies (vocabularies), and properties (structure) that describe concepts. Moreover, ontologies can be characterized according to the level of formality: highly informal (natural language), semi-informal, semi-formal, and rigorously formal. The ontology applied in this study is considered semi-informal, since it is expressed in a “restricted and structured form of natural language” [5].

The main contributions of this paper are the following. Context Studio is further developed for the end users of small-screen mobile devices. Navigating and information presentation are designed for small screens, especially for the Series 60 UI. The context ontology and vocabulary models enable the generating of user interface elements automatically and dynamically. The ontology hierarchy is transformed into a directory model representation for the user interface, allowing easy navigation. Extensions and modifications to the ontology vocabularies can be automatically updated into the UI. The rule model is utilized to allow systematic management and presentation of rules in the user interface. Moreover, the results of a paper model user evaluation of the user interface are presented to evaluate the chosen UI navigation logic and appearance.

The paper is organized as follows. Context ontology with an enhanced vocabulary model is introduced. The ontology structure and vocabularies are used in a formal rule model, which is presented next. The user interface model is based on the representation of vocabularies and rules. The navigation in the UI is explained with examples. Finally, the UI is evaluated based on the results from the paper prototype usability study, followed by future work and conclusions.

2. CONTEXT ONTOLOGY AND VOCABULARY MODEL

Context ontology facilitates automatic generation of graphical user interface views for triggers and actions in Context Studio, and lets users navigate within the ontology hierarchies and choose contexts for context-action rules. The ontology contains concepts that are used as triggers and actions in context-action rules. For example, the ontology can describe a set of possible contexts that can be produced by a classifier from sensor data. In an online system the classifier produces instances of the ontology, which are used for triggering the user-defined rules.

The ontology is defined by a knowledge engineer or a computer scientist before use, and it should preferably be verified by a group of people in order to make it a shared conceptualization. The end user of Context Studio utilizes the ontology through the graphical UI in defining context-action rules.

The context ontology applied in Context Studio consists of two parts: structure and vocabularies. Structure defines the common properties of context that are used across different domains and

applications. Vocabularies are application- or domain-dependent expandable context conceptualizations, which aim at understandability and simplicity for the end user as well as the application programmer. New vocabularies are developed for new domains.

The ontology structure is defined as a set of properties. Each context (object) is described using six properties: Context type, Context value, Source, Confidence, Timestamp, and Attributes [7]. Defining context vocabularies concerns defining sets of Context types and Context values. Other properties are not discussed in detail in this paper.

Ontology vocabularies are designed according to the domain or application needs. Vocabulary design is a process of defining human-understandable domain- or application-specific values for the Context type and Context value properties. Those values should categorize and describe the possible real-world situations so that the information is useful for applications and understandable by humans.

Context types are defined for naming and categorizing context values. Context type resembles a variable name, and context values resemble the set of values for the variable. Figure 1a illustrates the vocabulary model for one context type and a set of context values for the type. Relations between concepts are only named for clarity, and are not used for, e.g., inheritance. Figure 1b shows an example of one context description for the vocabulary. Context type can have one or more concepts. Each context type can have one or more context values. The context instance at a certain time can have one of the values for each Context type. Different Context types can have one or more common concepts. Hence it is possible and useful to form a tree hierarchy of Context type concepts, where the leaf nodes represent the context values. The hierarchy can be utilized in querying and subscribing to branches of the context tree, instead of only one path.

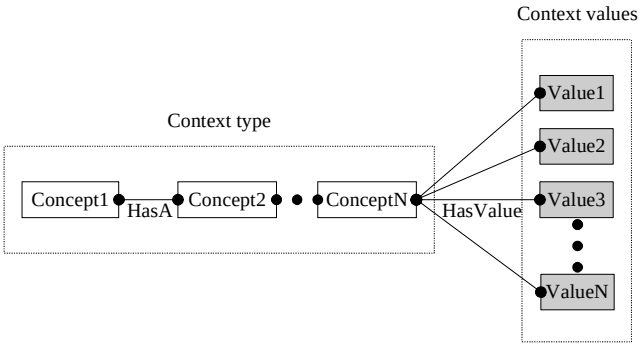

Figure 1a. A model for creating vocabularies consisting of Context types and Context values.

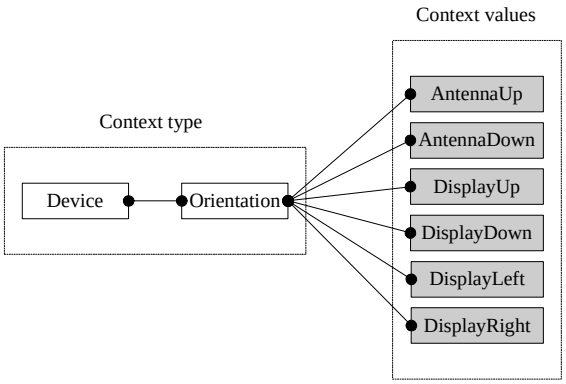

Figure 1b. An example Context type and a set of Context values.

The Context type concepts should be categorized from generic toward specific, so that more generic concepts are to the left, towards the root, and more specific concepts are to the right, toward the leaves. In this manner Context types can have common more generic concepts, but are separated by the more specific concepts.

For simplicity and ease of use, very long Context types should be avoided. At maximum, Context types should have three or four concepts. For example, if Context type concepts are modeled as folders in a user interface, navigating deep folder hierarchies is slow. On the other hand, Context types should be specific enough to allow having a small set of values. Too large a set of values is difficult to handle, at least in a user interface, if the user has to choose from a set of values. Each Context value should have a potential relevance to some application.

Table 1 presents an example of a context vocabulary describing device category contexts that are abstracted from acceleration and touch sensor data. The example in Figure 1b is described in text form in the first row of Table 1.

Table 1. Device category sensor-based context vocabulary example Context types.

Context type	Context values
Device: Orientation	AntennaUp, AntennaDown, DisplayUp, DisplayDown, DisplayRight, DisplayLeft
Device: Placement	AtHand, NotAtHand
Device: Activity	Still, Activity

Actions to be performed based on contexts can be represented by using the context vocabulary model. Actions are defined with two properties, Action type and Action value, which describe actions as Context type and Context value describe contexts. Table 2 presents an example of an action vocabulary. Moreover, external

devices can announce their actions, which can dynamically be included as Context Studio action vocabularies.

Table 2. Phone category action vocabulary example action types.

Action type	Action values
Phone: Applications: Profiles	Normal, Silent, Outdoors, Meeting
Phone:Joystick	JoystickUp, JoystickDown, JoystickLeft, JoystickRight, JoystickPress
Phone:Keypad	LockKeypad, UnlockKeypad

The ontology vocabulary model enables generating elements of the Context Studio user interface from the vocabularies. The hierarchical vocabularies are extensible and modifiable, and these qualities are inherited by the generated UI. Without the concept hierarchy, flat lists of Context values would easily grow too large to enable usable interaction.

3. NAMING CONVENTIONS

The context management of Context Studio is designed to utilize the context framework [8]. Appropriate naming is essential, particularly in describing the properties, Context type, Context value and source that are required for each context object added to the context manager blackboard. These properties are also used for accessing context information from the context manager. There are two main aspects to be considered in naming Context types. First, appropriate naming should reflect the meaning of context to the user, which is either the application developer or personalization tool user. Naming should reveal the use of the context. Second, correct naming convention ensures that the user of the context information can fully utilize the features provided by the context manager.

When Context types are built as paths consisting of elements from a generic to specific concept, context information can be accessed with partial Context types that represent a larger context information subset. This naming convention has also been utilized in CEP, where the reference to a subset of Context type hierarchy is called a wildcard [9]. Moreover, CEP recommends that vendor-specific context types are named starting with a prefix that names the vendor, e.g., “x-vendor_name:”, followed with the normal Context type definition.

The set of Context values can be either numerical or symbolic. If Context values are described as numerical, for application use, they should have an explicit meaning to humans, such as environment temperature in degrees. If raw numerical measurement values are used, naming a Context value set is not required. Context type can be used normally. If Context values are defined for the purpose of application personalization by the user, they should be understandable by humans and symbolic, and the number of values in the set should be low enough to allow choosing a value from the set to function as a condition in a rule. When a small set of values is required, numerical values can be divided into named intervals - e.g. temperature value can be “over 20”. Several other methods exist for abstracting numerical values into symbolic values [8].

4. CONTEXT STUDIO RULES

Context Studio enables the user to specify rules, which connect contexts to actions. Once the rules are active in the mobile device, contexts cause actions, as specified in the rule. Rule condition part elements are called triggers. A trigger can be any event or state that can be used for activating an action. Triggers are instances of the context vocabulary. Context type (variable name) and Context value (variable value) together define a trigger.

An action is any application, function, or event that can be activated when a set of triggers occur. The set may contain one or more triggers. A rule action part element is called action. For clarity, one rule may contain one action. Action type and Action value define an action.

A rule is an expression connecting a set of triggers to an action. Rules consist of one action element and one or more trigger elements. Rules are of the form IF trigger(s) THEN action. Based on the earlier user study [11], the rule structure is revised. To keep specifying rules simple for the user, the only operator for connecting triggers is AND. An example of a rule containing multiple triggers would look as follows: IF trigger1 AND trigger2 AND trigger3 THEN action. The operator OR is implemented by allowing the user to specify additional parallel rules for the same action. The user is allowed to make incomplete rules - i.e., the action or trigger part is allowed to be missing. Incomplete rules are represented with icons as disabled in the UI.

Context Studio adopts the CEP context script representation for rules [9,10]. Context scripts are semi-formal XML descriptions that can be used for describing rules. An example of a rule described as a context script is as follows.

```
<script
xmlns="http://www.nokia.com/ns/cep/script/1.0/"
xmlns:cep="http://www.nokia.com/ns/cep/1.0/">
<if>
<and>
<equal>
<atomRef name="Device:Activity" />
<cep:string>Still</cep:string>
</equal>
<equal>
<atomRef name="Device:Orientation" />
<cep:string>DisplayDown</cep:string>
</equal>
</and>
<actions>
<notifyApp
receiver="phone:apps/Activator">
<cep:atom
name="Phone:Applications:Profile">
<cep:string>Meeting</cep:string>
</cep:atom>
</actions>
</if>
</script>
```

The example rule causes the device profile to be set to Meeting if the device orientation is display down and the device is still. Rules conforming to the script specification and Context Studio rule form can be displayed in the user interface. Hence it is possible to use rules defined by others, and exchanging rule scripts over the air is feasible. However, rules can be personal and thus not usable by others. For example, if the location of person's home is bound to coordinates, the description is not valid with anyone else's home.

Context management in Context Studio is based on the blackboard-based context framework for mobile devices [8] and rule scripts are evaluated with an inference engine, called script engine, [9] which uses context manager. Condition operations such as And, Or, Not, Equal, Not Equal, Less, Greater, Less Or Equal, Greater Or Equal, and In Range are supported by the context script model and the inference engine. In an online system context manager receives all context information from context sources and abstractors, and indicates script engine upon changes in those contexts that are used in the rule scripts generated by the Context Studio that have been defined by the user with the graphical UI. Script engine evaluates the rules for the changed contexts, and fired rules are used for activating application actions or platform events as defined in the rule. Details of context, rule, and application action management relating to Context Studio will be published later.

5. USER INTERFACE

Context Studio rules are specified by the user with a graphical user interface, following the notion of the semi-automatic adaptation model. The user interface is an application of context ontology. The representation of triggers and actions is based on the ontology vocabulary model. The Trigger, Action, and Rule view elements in the user interface can be automatically and dynamically generated based on the ontology and rule models. The ontology hierarchy is transformed into a folder-file model representation in the user interface, allowing easy navigation. Context and Action type concepts are represented as folders and subfolders, according to the vocabulary hierarchy. Context and Action values correspond to files in the concept folders. Extending and modifying vocabularies is possible at runtime, and the UI can be updated correspondingly. New Context and Action types are presented as new paths in the UI, and new Context and Action values are presented as new files in the folders.

Figure 2 shows three phases of navigating in the triggers view of the Context Studio UI. The UI corresponds to the vocabulary example given in Table 1. The UI is designed for small size screens, and the style follows Nokia Symbian Series 60 guidelines.

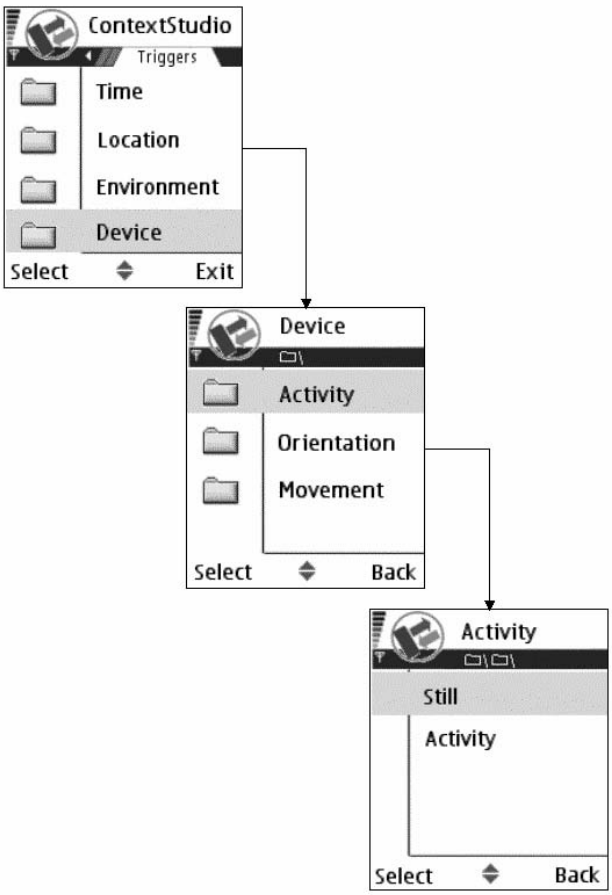

Figure 2. Navigating the triggers view in Context Studio.

In Figure 2 the user is navigating the Device category Context vocabulary in order to find the correct trigger for the rule in the earlier example. The user follows the path Device\Activity and, after choosing the context value Still, the view is returned to the rule edit view, Figure 3. Action hierarchy is displayed and navigated in a similar manner in the UI.

The rule scripts can be created, viewed and edited in the UI rule view. Figure 3 presents the view of the rule given in the example script.

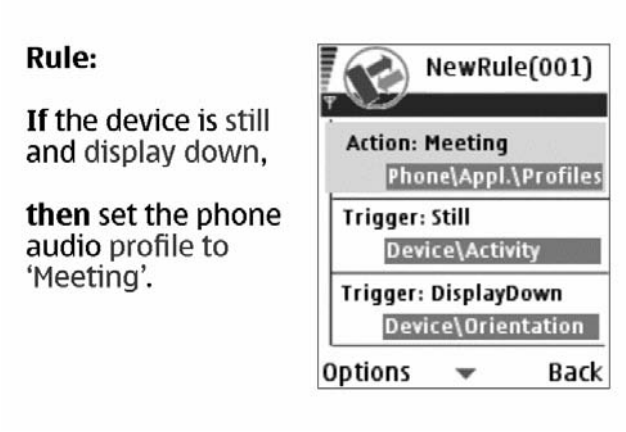

Figure 3. Context Studio rule edit view.

6. USER INTERFACE EVALUATION

The earlier Context Studio prototype [11] was evaluated with users. In this study the tool was improved based on the results from the earlier prototype, and redesigned to suit small-screen devices. Fewer operators were used for rule specifying, since overly complex rule structures have been found confusing by the users. The context and action navigation logic were redesigned and the rule view was simplified; the user may freely choose the trigger(s) for any rule, instead of marking predefined triggers. The hierarchy of contexts and actions is based on the scalable ontology vocabulary model, and the UI is generated accordingly.

6.1 Method and Participants

The Context Studio user interface was tested by paper prototyping with subjects who had no previous experience of context-aware systems. Each test session was conducted in controlled premises and lasted about 1.5–2 hours. A formative testing and evaluation approach was used for identifying problems and evaluating the overall design [1]. The test procedure was explained to the subjects and they were encouraged to voice their thoughts [12]. First, the subjects filled out a background questionnaire, charting their experience as mobile phone users. After this, they interacted with a paper prototype according to given scenarios, which included a short story and a task to be performed with the application. Between the scenarios the subjects were interviewed with questions related to each scenario. Finally, the subjects filled out a survey, where they assessed and commented on the test.

Eight subjects (4 male and 4 female), age 20-39, from different fields of study or work participated in the test. All eight subjects were users of mobile phones, five having previous experience with Nokia Series 60 phones. None of the subjects worked in the mobile phone industry.

6.2 Test Results

During the test the subjects constructed rules according to four scenarios, of which the first two are presented here as examples.

Scenario 1: *You often call your best friend Mike. You want to be able to make the call quickly and without watching the phone, as you call him many times while driving a car or riding a bike. The phone recognizes a 'shaking' movement. Now you want to determine that when you shake the phone it starts calling Mike.*

Scenario 2: *You quite often spend an evening in a nightclub with your friends. When you are clubbing you want your phone to turn to the 'Outdoor' profile so that you are able to hear when you receive a call or a message, and so you define an automated profile change.*

A rule was constructed by selecting Context and Action values in the rule edit window (Figure 3). The selected values were found by navigating in the triggers and actions folder hierarchy (Figure 2). When constructing the rules, all subjects started from the rule edit window, where the *Action* and *Trigger* controls initially contained the value 'none'. Seven of the eight subjects first defined *Action* and then *Trigger(s)*, whereas one subject (#6) committed the tasks vice versa.

The concepts *Rule*, *Action* and *Trigger* were, in general, well understood. With the first task, six subjects could instantly complete the rule without any problems. Two subjects had initial difficulties in constructing a rule. Subject #5 failed to complete a functional setting,, as (s)he defined only an action and no trigger. Subject #4 required some help with the application logic, although (s)he understood the idea of building a rule that would contain a condition and a consequence part. After completing the first task, none of the subjects failed to construct a rule, which suggests that the tool usage is easy to learn.

With the first scenario, four subjects (#1, 3, 6, 8) erroneously expected the gesture named 'shaking' to be the action as it described a physical action from the user. However, these subjects quickly spontaneously realized their misunderstanding when navigating in the action menu as the vocabulary referred to the consequential part of a rule.

The task given in Scenario 2 did not provide straightforward instructions for appropriate trigger selections but subjects had to infer them themselves. With this task the aim was to study the subjects' ability to define several triggers for one rule. Only two (#2, 8) subjects spontaneously realized that several triggers could be defined, and one of them (#8) still chose to use only one trigger. After completing the rule construction, the subjects were advised that defining several triggers was possible. However, they still chose not to add any more triggers but retained their original selections.

Generally, the vocabulary used in folder structure was perceived as intuitive and the navigation in the hierarchical structure did not cause difficulties. The presentation of both the trigger and action part in the same rule edit window (Figure 3) was perceived as a useful starting point for constructing a rule and for completing the editing. Some error steps appeared while navigating in the menu hierarchy. The errors were due to a misunderstanding with the vocabulary, leading to errors such as entering to the *InputEvents* folder instead of *Gestures (#8)*. Two of the subjects mentioned being confused because of the length of the hierarchical menus: *'I have now long ago forgotten where I have gone'* (#4), and *''These seem to be behind quite a many selections…'* (#7). Some questions related to vocabulary were raised if terms were not commonly used in the subject's field of work or study.

At the end of the test the subjects were asked to assess the test on a scale from 1 (lowest) to 5 (highest). The ratings given by the subjects for the following items were *Usefulness of selected*

functions 3.6 (standard deviation 0.52), *Appearance* 3.9 (0.64) and *Logical structure* 3 (0.76). When asked *'Do you feel you would benefit from the Context Studio application?'*, all subjects answered affirmatively. When interviewed, it was found that there was significant variation in which rules were perceived as most useful and which rules the users would probably use, depending on the user's lifestyle.

6.3 Analysis of the Test Results

The results of the user tests indicate that Context Studio is a potentially useful tool, and users could perform the given tasks well. The overall number of errors related to the interaction with the user interface was small. In particular, the performance with the vocabulary and hierarchical folder structure was good. Moreover, the idea of constructing a rule having a condition and a consequence part was well understood. Potential difficulties relate more to *the kind of rules* that are constructed than to the actual ability to do it.

Users have a tendency to avoid constructing long and complicated rules. The test indicates that this is affected by the user's low tolerance of required effort, elongated time spent with manual settings, and problems with outlining the increasing complexity.

Furthermore, the test results indicate that there are potential usability risks with users' subjective perceptions of context; they have varying opinions on the meaning of subjective concepts such as 'cold temperature'. Objective contexts should be preferred in the ontology vocabulary design [7]. The vocabulary should be verified with a group of people before use, and deep hierarchies should be avoided if possible.

The tests indicate that users are strongly influenced by what they perceive as useful device features for their particular lifestyle or usage preferences. The users would like to make personal context-based rules, which proves the usefulness of the chosen semi-automatic adaptation hypothesis.

7. CONCLUSIONS AND FURTHER WORK

Context Studio, an application personalization tool for semi-automated context-based adaptation, was further developed for the end users of small-screen mobile devices. The graphical user interface of Context Studio was designed to utilize context ontology. The ontology vocabulary hierarchy is transformed into a folder-file model representation in the UI. The ontology, with an enhanced vocabulary model, offers a scalable information representation and easy navigation of context and action information in the UI, and enables straightforward updating of the UI according to changes in the vocabularies. Furthermore, the ontology supports the utilization of context scripts as a formal rule model.

The tool was evaluated with paper prototyping. The usability study indicated that the tool was found useful, although the preferred context-aware features differed between the subjects. The general idea of constructing a rule with a condition and consequence part was quite well understood, and users had no problems in defining rules. However, they were not eager to construct rules matching multiple contextual conditions. Navigating hierarchical folder structures for selecting desired triggers and actions was intuitive, although care must be taken when designing vocabulary to keep it easily understandable and unambiguous.

A tendency to make simple rules was an obvious trend in the user test. Adding more rule operators is a consideration for the future, but the additional complexity of rules should not be allowed to reduce the usability of the tool. If making rules is too difficult, users will probably not use the tool.

Context Studio can be used for personalizing applications of a small-screen mobile device with a usable graphical UI. The range of possible context-aware features depends on the number and quality of the context values that can be acquired and abstracted from sensors and other sources, and the number of available actions. The usefulness of the self-configured features will be decided by the end user. Future work involves implementing Context Studio on top of the context framework in the target platform, and context abstractors to produce the contexts defined in the ontology vocabularies. The complete tool will enable study of the UI usability in the target device, and easily defining and evaluating context-aware application features in a real usage of a mobile handheld device.

8. ACKNOWLEDGMENTS

We thank Jani Mäntyjärvi and Urpo Tuomela for their helpful comments.

9. REFERENCES

[1] Carroll, J.M. Making of Use. Scenario-based design of human-computer interactions. (Chapter 8). The MIT Press 2000, 368 p.

[2] Dey, A., Hamid, R., Beckmann, C., Li, I., Hsu, D. a CAPpella: Programming by Demonstration of Context-Aware Applications. *Proc. CHI 2004*, ACM, 2004.

[3] Henricksen, K., Indulska, J., Rakotonirainy, A. Modeling Context Information in Pervasive Computing Systems. *Proc. International Conference on Pervasive Computing 2002*, LNCS 2414. Springer-Verlag, 2002, 167-180.

[4] Erickson, T. Some problems with the notion of context-aware computing. *Communications of the ACM*, 45(2), 2002, 102-104.

[5] Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O. Ontological engineering. Springer-Verlag, 2003, 403 p.

[6] Greenberg, S. Context as a dynamic construct. *Human-Computer Interaction*, 16, 2001, 257-268.

[7] Korpipää, P., Mäntyjärvi, J. An Ontology for Mobile Device Sensor-Based Context Awareness. *Proc. 4th International and Interdisciplinary Conference on Modeling and Using Context 2003*, LNAI 2680, Springer-Verlag, 2003, 451-459.

[8] Korpipää, P., Mäntyjärvi, J., Kela, J., Keränen, H., Malm, E-J. Managing Context Information in Mobile Devices. *IEEE Pervasive Computing Magazine special issue: Dealing with Uncertainty* 2(3), IEEE Computer Society, 2003, 42-51.

[9] Lakkala, H. Context Exchange Protocol Specification. 2003, 28 p. Available: http://www.mupe.net.

[10] Lakkala, H. Context Script Specification. 2003, 22 p. Available: http://www.mupe.net.

[11] Mäntyjärvi, J., Känsälä, I., Tuomela, U., Häkkilä, J. Context-Studio - Tool for Personalizing Context-Aware Applications

in Mobile Terminals. *Proc. Australasian Computer Human Interaction Conference 2003*, 2003, 64-73.

[12] Preece, J. Sharp, H, Rogers, Y. Interaction Design: Beyond Human-Computer Interaction. (Chapter 12) John Wiley & Sons, Inc. 2002, 519 p.

[13] Ranganathan, A., Campbell, R. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing Journal* 7, Springer-Verlag, 2003, 353-364.

[14] Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Laerhoven, K., Van de Velde, W. Advanced interaction in context. *Proc. 1st International symposium on handheld and ubiquitous computing 1999*. Springer-Verlag, 1999.

[15] Sohn, T., Dey, A. ICAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications. *Ext. abstracts CHI03*, 2003, 974-975.

[16] Wang, X., Zhang, D., Gu, T., Pung, H. Ontology Based Context Modeling and Reasoning using OWL. *Proc. Workshop on Context Modeling and Reasoning at IEEE International Conference on Pervasive Computing and Communication*, 2004, 18-22.