# Managing Context Rules in Mobile Devices

Waltenegus Dargie

Chair of Computer Networks, Faculty of Computer Science, Technical University of Dresden
D-01062 Dresden, Germany
waltenegus.dargie@tu-dresden.de

**Abstract**. An essential aspect of Mark Weiser's vision for ubiquitous computing is that computers become "invisible", demanding very little attention of their users. For this to happen, computers should be able to establish a shared understanding of the conceptual and social settings in which they operate, i.e., they should be context aware. This paper addresses one aspect of context-awareness, namely the management of context rules in mobile and wearable devices. Context rules are employed to predict higher-level human situations (activities), and to actuate desirable operations which are suitable to these situations. Because context rules are application specific, they are defined by application developers and users. So far, little consideration is given in the literature to take the habit (experience) of users into account to dynamically generate and manage context rules. In this paper we will provide a framework to associate decision events (signifying the actions performed by a user) with numerous context types which describe a situation of interest. From an aggregate of decision-context associations we generate context rules which enable mobile devices to proactively provide useful services.

## 1. Introduction

A little over a decade ago, Mark Weiser asserted that in less than 20 years, ubiquitous computing would become dominant, and computers will cease to be obtrusive. Moreover, a single user would interact not just with one, but with many devices, each of which is "so imbedded, so fitting, so natural, that we use it without even thinking about it [1]". Since then, indisputable achievements have been made to substantiate his assertion: Whereas once it has been considered to be a revolution to provide individuals with personal computers, it is now an everyday practice to carry with us several mobile devices and communicate with them on the spur of the moment [2]. The devices with which we interact are, however, everything but invisible; we still switch off our mobile phones or set their ringing style to vibration mode manually whenever we attend meetings, lectures, etc., or forget to do so and end up upsetting others as well as ourselves. Furthermore, despite significant advances in the communication and networking technologies as well as interactive office and home appliances, by and large, human intervention is required to decide whether and how our devices should cooperate with each other. Subsequently, the increment in the ratio of mobile devices to a user causes distraction where it should make everyday life more manageable [3].

If mobile and wearable devices are able to capture what is taking place around them (i.e., if they become context-aware), they can provide suitable services proactively, and adapt to their surrounding autonomously.

Context-awareness can be achieved in two stages: First, the desired context of a user or a device needs to be captured; second, an operation corresponding to the context of interest needs to be specified. Both process entail the definition and management of a set of context rules, which are useful (1) for inferring higher-level human situations (activities) from numerous explicit context types that can directly be captured by employing sensors [4, 5]; and (2) for actuating certain operations inside mobile devices whenever a context of interest is captured and a condition is evaluated to be true [6].

In this paper, we will discuss the role of context rules in detail and propose a framework that enables mobile devices to dynamically generate (at least in part) context rules by associate a user's activities with a context in which the activities unfold. The decision-context associations are eventually employed to dynamically generate context rules.

The rest of this paper is organised as follows: in section II, we will discuss the way context rules can be organised in context-aware systems and applications; in section III, we will summarise related work; in section IV, we will consider three different scenarios for which the dynamic generation of context rules applies; in section V, we will discuss decision-context associations as well as event expression semantics; in section VI, we will discuss the components of the architecture, followed by experiment results in section VII. Finally, in section VIII, we will close by providing concluding remarks and future work.

## II. Context Rules

Context rules are essential components by which certain criteria are evaluated to determine how mobile devices should adapts to a context. Figure 1 shows the way rules are embedded into context-aware applications. Because context rules are application-specific, they are usually specified either at design time by the application developer or at runtime by the user of the application. User-defined rules are often simple rules describing simple behaviour, since user side operation should avoid excessive cognitive load for the application to be relevant to the user.

Defining context rules at design time and embedding them into an application's business model ensures a side-effect free modification of behaviour. Moreover, it enables the designer to specify a mechanism (identify the types of sensors) for capturing the explicit context types. This approach, however, should be complemented by a learning scheme so that the system can accommodate new context types which are not expressed in a rule at design time, but may as well reveal some vital aspects of a situation of interest. This is particularly true if there are multiple ways of capturing complex human situations, such as human activities or emotional states [7].

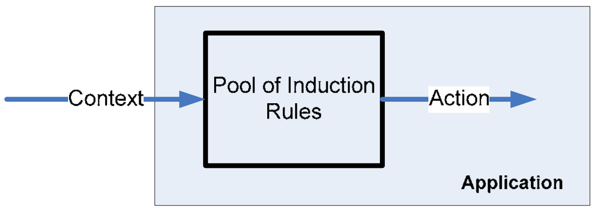


Figure 1: A context rule definition and processing component as a part of a context-aware application.

If all exhaustive context types cannot be specified at design time, the system should be able to accommodate new aspects by learning about their capability to describe a situation of interest. Besides identifying the context types, rules also require the specification of existential quantifiers which are used to set criteria or thresholds. Consider the following example: 'If environment loudness is above 12 db, set ringing tone volume to 2.5' [8] – here the context type is environmental loudness, and its value is 12 db. To define such rules, the existential quantifiers 12 db and 2.5 (whatever the unit) require expert-knowledge. On the other hand, if a context-aware system is capable of learning new environments, it may be able to dynamically compute the threshold values associated with a context.

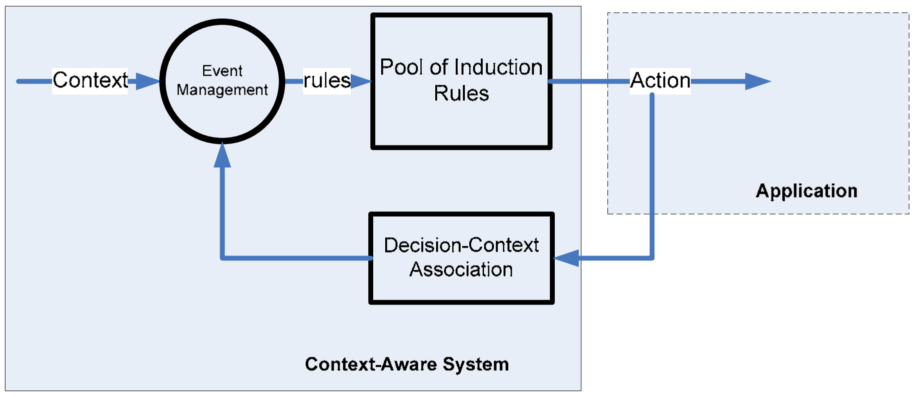


Figure 2: Use of context-decision association to dynamically generate context rules.

We modify the architecture of figure 1 in order to allow mobile devices to learn about new aspects of a situation of interest, and to produce context rules which determine the way mobile devices should dynamically adapt to the situation. To this end, we introduce a feedback system which associates a user's decision (signifying a desirable modification of a behaviour inside a mobile device) with a context (representing the situation of interest), and which eventually generates context rules from the association. Moreover, we introduce the decision-context association component and the event management component to manage context rules. Furthermore, we take out the

rule organiser (managing a pool of induction rules in figure 1 and 2) outside of a context-aware application.
In this paper, an application is defined to be a software entity which automates a process that should otherwise be carried out manually by a user. Some examples are word processors and tourist guide applications. A context-aware system, on the other hand, is responsible for managing context and context rules, so that an application can proactively provide a useful service.

## III. Related Work

Ranganathan et al. propose a rule description and composition framework based on a first order predicate logic [9]. A context is represented as a predicate with three arguments: subject, relater, and object. The subject part of a context predicate refers to an entity to which the context refers; the relater refers to the way the subject is related to the object; and the object part of a context predicate refers to a value (state) of the context predicate that is associated with the subject. The model enables the creation of complex operations using Boolean operations and quantifiers. Similarly, Wang et al. [10] propose the Semantic Space framework in which a logic-based reasoner employs a Generic Rule Language based on the Jena2 framework to perform forward- and backward-chaining reasoning about human activities and whereabouts. In both cases, however, rules are created manually, by application developers.
Castro and Muntz [11] propose a framework for context-based recording and retrieval of very large multimedia objects in an interactive environment. The framework enables the dynamic association of geographical locations during the recording of a place of interest, so that a context-based query is possible with minimal signal-processing. Likewise, Pascoe [12] proposes a framework for associating location information with a note to enable context-based classification and retrieval of data. For example, a field worker in Kenya creates a note whenever he observes a wild animal; a context-aware system observing the creation of a note will internally associates the note with the output of a GPS receiver. Once an association is made, the field workers can query the system to learn about which wild animals are found in a certain area. We build on the experiences learned from the approaches of Castro and Muntz as well as Pascoe; our contribution is that we associate several context types besides location information. Moreover, our architecture enables us to support multiple applications.

## IV. Scenarios

A wearable system can be trained to generate context-rules by associating:

1. The decision events of a single application with a specific situation;
2. Multiple decision events of a single application with various situations; and,
3. Multiple decision events of multiple applications with a specified situation.

An example for (1) is training a system to switch a mobile phone to a vibration mode whenever a user attends a particular lecture. The task of the system is to recognise the occurrences of the context types describing the lecture session and to associate the set of context types with the switching of the mobile phone into a vibration mode. In (2), the goal is to train a system to associate different context types to a set of decision events which originate from a single application only. This happens, for example, when a user trains a system to modify the configuration of a mobile phone in different situations. The various decision events correspond to (i.e. refer to) the actions which switch the mobile phone to a vibration mode when the user attends a lecture; adjust the volume when the user walks along a noisy street, and so forth. Here the system should recognise the occurrences of the various decision events, and should associate each decision event with context types which represent the corresponding situations of interest. In (3), the goal is to train a system to recognise the activities of multiple applications which take place in one and the same situation. A typical example is the set of related activities which take place during a presentation session. The activities may include: adjusting a room's light system, turning on a beamer, loading a Power-Point application, and so on.

## V. Decision-Context Association

We associate a user's decisions with a situation in which the decisions are made. If the decisions are habitual decisions, then the systems identifies a set of context types which best describe the situation in which the decisions are made. When next the context types are captured, the system executes actions with which the context types are associated. If the system associates wrong context types with a decision, a user inputs a negative feedback, prompting the system to disintegrate the wrong decision-context association, a process we labelled as *unlearning*.
To manage decision events, there needs to be a formal expression of events and of context rules. We distinguish between primitive events and composite events. Primitive events are those predefined in the system, and a mechanism for their detection is assumed to be available. Primitive events include temporal events as well as events generated by the invocation of methods or subroutines to perform specific actions, i.e., decision events. All physical aspects which are captured by physical sensors (temperature, relative humidity, light intensity, etc.) are described as primitive context events. In general, a primitive event, *E*, is expressed by an event expression, which includes the event's name, the subject to which the event refers, the event's value and the time of occurrence (timestamp). Hence, a primitive event is represented as a predicate with four arguments. The semantics for expressing a primitive event is given by:

$$E(t) \equiv (\exists n)\,(\exists s)(\exists v)(\exists ts)(event(n,\ s,\ v,\ ts) \wedge (t = ts)) \tag{1}$$

The existential quantifiers: *n*, *s*, *v*, and *ts* are the event name, the subject, event instance (value), and the timestamp, respectively. Note that *v* in equation (1) can be a numerical value or an event object. Equation (2) and (3) demonstrate a context and decision event, respectively.

$$E_{room}(t = 10{:}00\,AM) \equiv event(temperature, room5052,\ 20°C,\ 10{:}00\,AM) \tag{2}$$

$$E_{mobile}(t = 10{:}05\ AM) \equiv event\ (switch,\ \ nokia6275i,\ vibration,\ \ 10{:}05\ AM) \tag{3}$$

Equation (2) describes the temperature of a room at 10:00 AM, while equation (3) describes the occurrence of a decision event after a Nokia 6275i mobile phone's ringing style is changed to a vibration mode. Once primitive events are described this way, we can define a context rule using first order predicate logic. For example, to evaluate a temperature measurement of *room5052,* and to adjust a heater accordingly, the rule expression is given by equation (4), which states that if the temperature measurement[1] of room5052 goes below the threshold *j*, heater *k* should be adjusted such that the value of *k* is between *l* and *m*.

$$(\forall t)(\forall i)(\forall k)\ (\exists j)(\exists l)(\exists m)\left( E_{room}(t) \wedge (i < j) \supset heater(k) \wedge (l \le k \le m) \right) \tag{4}$$

## A. Observation Time

The Primitive events discussed so far are useful for modelling simple behaviours. However, for expressing complex behaviours, it is necessary to detect certain combinations of different events as a single event, i.e., as a composite event. Composite events are defined by applying event operators to constituent events (i.e., primitive events), and are useful for expressing complex behaviours. One of these behaviours is an observation time, a period required by a context-aware system to identify those decisions which are habitually made when a mobile user is in a situation of interest. It can be ether an absolute temporal event or a relative temporal event. An absolute temporal event corresponds to a unique time span on the time line with a clearly defined reference time and an offset time. A relative temporal event begins at a unique time on the time line, but the ending time depends of the frequency of occurrence of a specified decision event. We adopt two composite event expressions (*ANY* and *Aperiodic*) from the Snoop event expression language [13] to specify an observation time. The *ANY* composite event is a conjunction event that occurs when *m* out of *n* primitive events occur, regardless of their order of occurrence. Formally,

$$ANY(m, E_1, E_2, ..., E_n)(t) = (\exists t_1)(\exists t_2)...(\exists t_{m-1})\begin{pmatrix} E_i(t_1) \wedge E_j(t_2) \wedge ... \wedge E_k(t_{m-1}) \wedge E_l(t) \\ \wedge (t_1 \le t_2 \le ... \le t_{m-1} \le 1) \\ \wedge (1 \le i, j, ..., k, l, \le n) \\ \wedge (i \ne j \ne ... \ne k \ne l) \end{pmatrix} \tag{5}$$

[1] Equation (2) is used as a reference, where the temperature variable *i* is used instead of 20°C.

The composite event expression *Aperiodic* is the occurrence of an event, $E_2$, within a closed time interval $[E_1, E_3]$. Formally,

$$A(E_1, E_2, E_3)(t) = (\exists t_1)(\exists t_2)\begin{pmatrix} E_1(t_1) \wedge E_2(t) \\ \wedge (t_1 \leq t) \wedge ((t_1 \leq t_2 < t) \supset \neg(E3(t_2))) \end{pmatrix} \tag{6}$$

The *Aperiodic* composite event is a non-cumulative event; i.e., it is signalled every time $E_2$ is detected within the time interval started by $E_1$ and ended by $E_3$. In equation (6), $\neg E_3(t_2)$ denotes the non-occurrence of the event $E_3$ at time $t_2$.

Referring to the three scenarios we describe in section II, we will define three different observation periods, one for each scenario. Let $E_1(t) = t_1$, and $E_3(t) = t_2$. The observation time for training a system when to switch a mobile phone to a vibration mode is defined as an Aperiodic composite event, where $E_2(t)$ is defined as:

$$E_2(t) \equiv (\exists ts)(event_{mobile}(switch,\ nokia6275i,\ vibration,\ ts) \wedge (t = ts)) \tag{7}$$

The ANY composite event (where $m = 1$) is used to define the observation time for training a system to associate the decision events of a single application to different situations (scenario 2). Hence, $E_2(t)$ is given by $ANY(1, E_x, E_y,\ldots, E_z)$, where $(E_x, E_y,\ldots, E_z)$ are the various decision events which can be generated inside of a single application. On the other hand, to train a system to manage multiple applications in a similar situation of interest, such as managing a presentation session, a decision table is required to define all decision events which can be generated by the applications. Subsequently, the event enumeration of the ANY composite event includes all the events described by an event table.

## VI. Architecture

As can be seen in figure 2, our architecture has three components for associating decision events with context types, and for eventually generating and managing context rules. These are the *event management component*, the *rule organiser*, and the *decision-context associating component*. Interaction between the components depends on whether the system is in an observation phase or in an execution phase. During an execution phase, the responsibility of the system is to sense context types which signify a situation of interest, and evaluates the pool of induction rules to execute desirable actions.

### A. The Event Management Component (EM)

The EM has two essential tasks: (1) During an observation time, it queries context sources and pushes the result to the Decision-Context Association component, so that

the latter can associate decision events with a set of context types; during an execution period, it subscribes to context sources, and when a desirable context is captured, notifies the Rule Organiser about the result, so that a rule or a set of rules referring to the context type can be processed; (2) Immediately after an observation period is over, it receives a set of decision-context associations from the Decision-Context Association component to generate context rules.
At present, we have classified context types into three classes for the EM to carry out the second task: temporal, numeric, and nonnumeric context types. Examples of non-numeric context types are calendar entries (meeting, lecture, presentation, etc.), emotional states (stress level, etc.), mappings of an RFID (room A, room B, etc). Examples of numeric context types are temperature, relative humidity, sound pressure, etc. Since numeric context types are very difficult to model (they can assume an arbitrarily large range of values), we employed a heuristic decision to map them into a meaningful abstraction. For example, temperature measurements are mapped to *cold, lukewarm, warm, or hot*; relative humidity measurements are mapped to *dry, moderate, or moist*; sound pressure measurements are mapped to *silent, normal, loud, or noisy*, etc. We employed heuristic judgements because the meanings of the conceptual abstractions depend on the entity they describe. To evaluate a temporal context, the day of the week and the time of day are extracted from it. To determine existential components of a rule, for numerical contexts, we use standard deviation, for non-numerical components we use frequency of occurrence. Thus, the Event Management component attempts to learn patterns in a user's decision.
In general, when the EM receives decision-context associations, it identifies the most representative context types as well as the corresponding values from a decision-context association. These context types, joined by the conjunction operation, become the antecedents of a rule while the decision event, correspondingly the action referring to it, becomes the consequent of the rule. We will demonstrate this by example in section V1.

### B. The Decision-Context Association Component (DCA)

The DCA is responsible for subscribing to a single or multiple applications, for associating decision events with context types which are acquired by the system from the computing environment at the time the decision events are produced. As we have mentioned earlier, decisions are the basic elements with which useful services are executed or the behaviour of an interactive system is modified. When a decision event first occurs, it receives a UUID to identify it distinctively. We denote an instance of a decision event, $E_j$, by $e^i_j$, where $i$ indicate the relative time of occurrence of the event $e_j$ with respect to the occurrences of the same event $e_{j-1}$. A decision event, associated with contexts, is persisted by the DCA until an observation time is over. When an observation time for a given decision event is over, an aggregation of decision-context associations will be passed over the EM which processes the decision-context associations in order to generate context rules.

### C. The Rule Organiser (RO)

The OR manages context rules. It receives from the EM notification of the occurrence of a context of interest, and evaluates all rules referring to the context. The rules may deal with context inference, i.e., reasoning about a higher-level context which abstracts a real-world situation such as human activity, or actuation of a service, for example, switching a mobile phone to a vibration mode if the user happens to be attending a lecture.

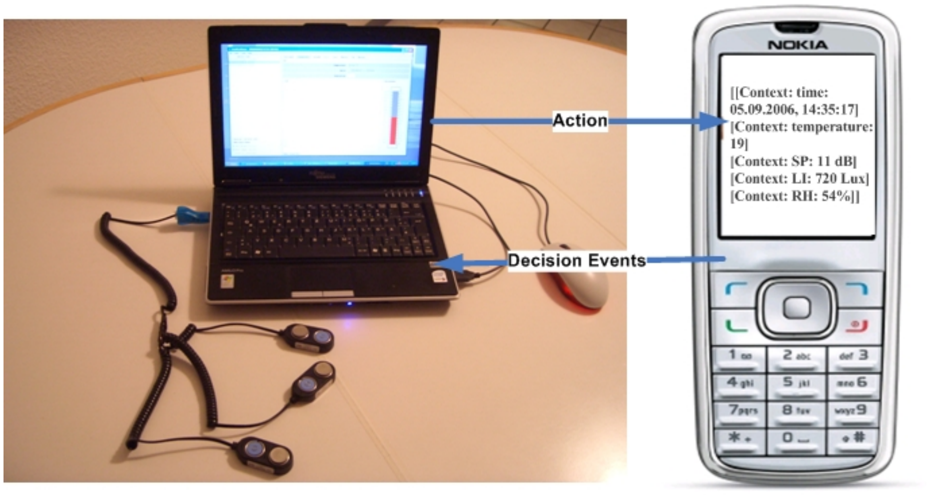
At present, the rules for context inference are described manually while the rules for service actuation are learned. We use Bayesian Networks to encode conditional dependencies between an implicit context (an abstraction of a conceptual or social situation) and several explicit context types which arrive from sensor abstracting components via the EM[2]. For example, given measurements from a temperature sensor, humidity sensor, and light intensity sensor, the Bayesian Network can determine whether the measurements refer to a room, a corridor or an outdoor place. We employ empirical knowledge to determine conditional dependencies and to setup the Bayesian Networks. Thus, the task of the RO as far as context inference is concerned is to receive sensor measurements from the EM, map the measurements according to a predefined rule to corresponding features (for example, a temperature measurement is mapped to *cold*, *lukewarm*, *warm*, or *hot*), determine conditional dependencies of the features to potential higher-level contexts, and finally, compute posterior probabilities to determine the most likely higher-level context. The second task of the RO, which is carried out autonomously, is to proactively actuate a service by associating the service with a context.

## VII. Validation

At present, our implementation of the architecture is capable of associating decisions events of a single application with simple but numerous context types each of which describes a situation of interest (scenario I). Work is on progress to support scenario (2) and (3).

We used the Nokia Mobile Internet toolkit 4.0 (NMIT 4.0), Series 40 Platform 3rd Edition, to simulate an application which autonomously adjusts the ringer style of a Nokia 6275i mobile phone. Our choice of the mobile phone is motivated by the richness of decision events which can be generated inside it. To support decision-context association, we integrated an Event Import and Export (EIE) component into the application, residing on the mobile phone. It is responsible for dispatching and receiving decision events from and to the mobile phone. During an observation time, decision events are exported from the application to the DCA (which runs remotely on a laptop); during an execution time, decision events (or more precisely the issuance for their creation) are imported from the rule organiser to the application to perform an action associated with the occurrence of a context of interest.

[2] Context reasoning is treated in detail in a separate paper [5].



```
[[Context: time:
05.09.2006, 14:35:17]
[Context: temperature:
19]
[Context: SP: 11 dB]
[Context: LI: 720 Lux]
[Context: RH: 54%]]
```

Figure 3: The 1-wire Sensor Network and the DCA Communication

To gather surrounding information, we employed the 1-wire[3] protocol to interface several Dallas semiconductor sensors (temperature, light intensity, relative humidity, etc.) to a laptop via the DS9490R USB adapter. The number of nodes in the 1-Wire network depends on the available sensors. We varied the number of active nodes at random to simulate dynamic availability of sensing elements. The Primitive Context Server is responsible for listening to the arrival and departure of an iButton® sensor, and reacts accordingly: when a sensor arrives, it creates an appropriate container to read data from the sensor, to attach a timestamp to the reading, and to store the result locally or notify the EM about the context. The context types which appear in a context-decision association are those the timestamps of which match the timestamp of the decision event.

We trained our system to autonomously *switch* the mobile phone (the decision event to be learned) whenever a mobile user attends a lecture session. An observation time of one month (an absolute temporal event) was set, during which time 4 lecture sessions were attended. The set of context types which were collected at the times the decision event was produced are enumerated in table 2. The context types which appear in the association most frequently (more than 50% of the time) are time, temperature, light intensity, and relative humidity. These context types are therefore chosen to be representative. The second step is to determine their values (or the features to which these values refer). This is *warm* for temperature; *moderate* for relative humidity; and *visible* for light intensity. The EM resolved the temporal context to day (*Tuesday*) and to the time of the day (14:30 {±5 minute}). The context types are used both for inference and for proactively actuating a service, i.e., switching the mobile phone to a vibration mode. The implicit context that was inferred by the Bayesian rule organiser is the higher-level concept (context) *room*. The resulting context rule is given by equation 7.

| | |
|---|---|
| 1 | [[Context: time: 05.09.2006, 14:35:17]<br>[Context: temperature: 19°C]<br>[Context: SP: 11 dB]<br>[Context: LI: 720 Lux]<br>[Context: RH: 54%]] |
| 2 | [[Context: time: 13.09.2006, 10:10:23]<br>[Context: RH: 48%]<br>[Context: SP: 11.98 dB]<br>[Context: temperature: 17°C]] |
| 3 | [[Context: time: 19.09.2006, 14:27:45]<br>[Context: RH: 52%]<br>[Context: temperature: 21°C]<br>[Context: LI: 1030 Lux]] |
| 4 | [[Context: time: 26.09. 2006, 14:32:10]<br>[Context: temperature: 23°C]<br>[Context: LI: 900 Lux]<br>[Context: RH: 48%]] |

TABLE 2: Decision-Context Associations.

$$(\forall d)(\forall t)(\forall l)\left(\begin{array}{l}\left(\begin{array}{l}day(d)\wedge time(t)\wedge loaction(l)\wedge \\ (d="Tuesday")\wedge between("14:25",t,"14:35")\wedge(l="Room")\end{array}\right)\supset \\ (decision(nokia6275i(vibration)))\end{array}\right) \tag{8}$$

Identification of representative context types describing a situation of interest is based on frequency of occurrence. This, however, is not optimal to model complex situations. The second version of our work employs the Decision Tree algorithm, which is more robust, and enables to associate decision events which arrive from multiple applications. This will enable us to model multiple activities which take place in one and the same situation.

## VIII. Conclusion

This paper was motivated by the hypothesis that if context-aware systems are capable of associating the activities of a user with the situation in which these activities take place, then they can proactively provide services suitable to the situation of a mobile user. Subsequently, we proposed a feedback system to context-aware applications, and introduced three components to the feedback systems in order to process decision events (signifying a user's activities) and context information. These components are the event management component, the decision-context associating component, and the rule organiser.

The event management component has two tasks. The first task is to query available context sources, so that the query result can be associated with a decision event, while the second task is to generate a context rule from an aggregate of decision-context associations. The decision-context associating component is responsible for subscribing to one or multiple applications to be notified of the occurrences of decision events whenever a user interacts with the application(s) to perform certain actions. The rule

organiser is responsible for reasoning about an implicit context from several explicit context atoms, and for firing context rules corresponding to the context of the user.
We could be able to train our system to dynamically load a relevant document whenever a user attended a lecture. The context types which provided indirect evidence about a lecture session were time, temperature, relative humidity, and light intensity. Obviously, these primitive contexts are not sufficient to describe a lecture session as accurately as possible. Our aim was, however, to identify representative contexts types based on frequency of occurrence instead of collecting as much context information as possible to model a higher-level context (lecture). Work is in progress to employ more powerful learning schemes such as hidden Markov models in order to model complex activities and situations.

## References

1. Weiser, M: The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.* 3, 3 (Jul. 1999), 3-11.
2. Dargie, W.: A Distributed Architecture for Computing Context in Mobile Devices. Doctoral Thesis, 2006.
3. Henricksen, K., and Indulska, L.: Modeling and Using Imperfect Context Information. In *Proc. of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops* (March 14 - 17, 2004). PERCOMW. IEEE Computer Society, Washington, DC, 33-37.
4. Korpipää P, Mäntyjärvi J, Kela J, Kernen H., and Malm E-J. 2003, Managing Context Information in Mobile Devices. IEEE Pervasive Computing 2, 3 (July-Sept. 2003), 42-51.
5. Schmidt, A., Beigl, M., and Gellersen, H.-W. 1999. There is more to context than location. Computers and Graphics 23 (1999), no. 6, 893–901.
6. W. Dargie, “Dynamic Generation of Context Rules,” Lecture Notes in Computer Science, Volume 3996, Jun 2006, Pages 102 – 115.
7. Korpipää, P., Koskinen, M., Peltola, J., Mäkelä, S., and Seppänen, T.: Bayesian approach to sensor-based context awareness. Personal and Ubiquitous Comput. 7, 2 (Jul. 2003), 113-124.
8. Mäntyjärvi, J., Himberg, J., and Huuskonen, P.: Collaborative Context Recognition for Handheld Devices. In: Proceedings of the First IEEE international Conference on Pervasive Computing and Communications (March 23 - 26, 2003).
9. Ranganathan, A., Campbell, R.: An infrastructure for context-awareness based on first order logic. Personal Ubiquitous Comput. 7, 6 (2003)
10. Wang, X., Dong, J. S., Chin, C., Hettiarachchi, S., Zhang, D.: Semantic Space: An Infrastructure for Smart Spaces. IEEE Pervasive Computing 3, 3 (Jul. 2004), 32-39.
11. Castro, P. and Muntz, R.: Managing context for smart spaces, IEEE Personal Communications, vol.7, no. 5, October 2000
12. Pascoe, J., Ryan, N., Morse, D.: Using while moving: HCI issues in fieldwork environments, ACM Trans. Comput.-Hum. Interact. 7, 3 (Sep. 2000), 417-437.
13. Chakravarthy, S., Krishnaprasad, V., Anwar, E., and Kim, S: Composite Events for Active Databases: Semantics, Contexts and Detection, in Proceedings of the 20th international Conference on Very Large Data Bases, September 12 - 15, 1994.