# Recognizing and Predicting Context by Learning from User Behavior[1]

*Rene Mayrhofer, Harald Radi, Alois Ferscha*
*Institut für Pervasive Computing*
*Johannes Kepler Universität Linz*
*Altenbergerstraße 69, 4040 Linz, Austria*
*email : {mayrhofer,radi,ferscha}@soft.uni-linz.ac.at*

**Summary**

*Current mobile devices like mobile phones or personal digital assistants have become more and more powerful; they already offer features that only few users are able to exploit to their whole extent. With a number of upcoming mobile multimedia applications, ease of use becomes one of the most important aspects. One way to improve usability is to make devices aware of the user's context, allowing them to adapt to the user instead of forcing the user to adapt to the device. Our work is taking this approach one step further by not only reacting to the current context, but also predicting future context, hence making the devices proactive. Mobile devices are generally suited well for this task because they are typically close to the user even when not actively in use. This allows such devices to monitor the user context and act accordingly, like automatically muting ring or signal tones when the user is in a meeting or selecting audio, video or text communication depending on the user's current occupation. This article presents an architecture that allows mobile devices to continuously recognize current and anticipate future user context. The major challenges are that context recognition and prediction should be embedded in mobile devices with limited resources, that learning and adaptation should happen on-line without explicit training phases and that user intervention should be kept to a minimum with non-obtrusive user interaction. To accomplish this, the presented architecture consists of four major parts: feature extraction, classification, labeling and prediction. The available sensors provide a multi-dimensional, highly heterogeneous input vector as input to the classification step, realized by data clustering. Labeling associates recognized context classes with meaningful names specified by the user, and prediction allows forecasting future user context for proactive behavior.*

**Keywords**
*Feature Extraction, Context Awareness, Context Prediction, Proactivity, Framework*

## 1. Introduction

Computing environments are changing rapidly, and the pace of this change is currently increasing. Due to broad availability of computing and network infrastructure, the potential audience of computing, communication or other services of informational nature is growing steadily. As a consequence thereof, ease of use becomes a primary concern.

The purpose of our study is to enhance *information appliances* [27] to predict context and deliver proactive services to the user. An information appliance is a device designed to perform a specific function, specialized in information, with the ability to share information with other appliances. They are currently implemented as, for instance, mobile devices or within Pervasive Computing.

Many have already presented their visions of future computers, including Mark Weiser with Ubiquitous Computing [39] (which is also called Pervasive Computing), Steve Mann with Wearable Computing [22], Hiroshi Ishi with Tangible Bits [18] and Hans-Werner Gellersen with Smart-Its [13]. Common to most of them are the paradigms of Mobile Computing and Context Awareness [35]. Although these visions are radically different, they all agree that user interfaces should become less obtrusive and "smarter" with regards to adapting to the user. Today, most interfaces are explicit ones, forcing the user to adapt to the interface, to learn how to use it. If a "Personal Computer" or "Personal Digital Assistant" (*PDA*) would live up to its name, it

---

[1] This article is an extended version of:
R. Mayrhofer, H. Radi, and A. Ferscha. Recognizing and predicting context by learning from user behavior. W. Schreiner G. Kotsis, A. Ferscha and K. Ibrahim, editors, *The International Conference On Advances in Mobile Multimedia (MoMM2003)*, volume 171, pages 25–35. Austrian Computer Society (OCG), September 2003.

should instead adapt to the user, offering implicit, intuitive and sometimes invisible interfaces.

Our work strives to add another aspect to the vision of future computers: proactivity. We postulate that a PDA, which is not bounded to being a single physical device, can only fulfill its intentions if it acts proactively – good human assistants stand out for this reason. Our idea is to provide software applications not only with information about the current user context, but also with predictions of future user context. When equipped with various sensors, an information appliance should classify current situations and, based on those classes, learn the user's behaviors and habits by deriving knowledge from historical data. Our current research focus is to forecast future user context by extrapolating the past.

It should be pointed out that the topic of proactivity in computer science is a controversial one, especially in HCI; the general concept of context awareness itself has to be handled with care [8]. Because the estimated current or predicted future context and thus implicitly the actions started by the appliance based on these assumption might be erroneous, they might need to be reverted by the user, possibly causing severe problems. Thus, proactivity in applications, when utilized for controlling actuators with impact on the real world, must be handled with care. However, the possible uses for predicted user context in applications are manifold, ranging from simple reminder messages to emergency procedures being started before an accident happens. Our work is primarily concerned with techniques that enable proactivity in embedded devices and leaves decisions about starting actions to applications built on top of it.

In the following, we present our architecture for an application framework which provides predicted user context on the basis of historical data in real-time. The remainder of this article is structured as follows: Section 2 defines our notions of proactivity and context. Section 3 lists related work and sets our work in relation to other projects. The main part of this article, section 4, explains the architecture including our contribution of adding proactivity to context awareness. In section 5, our implementation of this architecture in form of a cross-platform software framework is discussed and current results are presented in section 6. Finally, section 7 shortly summarizes our work and describes future aims.

## 2. Definitions

### 2.1 Proactivity

The term proactivity has been used in computer science mostly for software agents, where one important difference between agent oriented programming and object oriented programming is the proactivity of software agents [40]. Formally, the difference between reactivity and proactivity lies in the dependence of the current system output on the system state trajectory. If interpreted as an abstract (Moore) state machine, the internal "state" of a system at time $t$ can be described as $q_t = d\left(q_{t-1}, a_{t-1}\right)$, where $q_t$ is the current state, $q_{t-1}$ is the last state and $a_{t-1}$ is the input value at time $t-1$ (cf. [29]). In this definition, system inputs and state transitions are assumed at discrete time steps $t \in N_0$. The system output depends on the state – this is how the difference between reactivity and proactivity is defined in the context of this article. In a reactive system, the output $b_t$ at time $t$ only depends on the current and – implicitly – on past states:

$$b_t = l\left(q_t\right)$$

In a proactive system, it can also depend on predicted future states:

$$b_t = l\left\langle q_t, \overline{q}_{t+1}, \overline{q}_{t+2}, \ldots, \overline{q}_{t+m}\right\rangle$$

The future states $\overline{q}_{t+1}, \overline{q}_{t+2}, \ldots, \overline{q}_{t+m}$ for $m$ discrete time steps are predicted recursively by some arbitrarily complex process $\overline{q}_{t+i} = p\left\langle q_t, \overline{q}_{t+1}, \ldots, \overline{q}_{t+i-1}\right\rangle$; if only $q_t$ is necessary for predicting any $\overline{q}_{t+i}$, then $p$ can simply ignore the predicted states $\overline{q}_{t+1}, \ldots, \overline{q}_{t+i-1}$.

### 2.2 Context

Context has been defined by Dey [7] as

> *any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or a physical or computational object*

which we adopt in this article. A good overview on different definitions of context can be found in [34].

Describing the situation in general, context can have many aspects, typically:

- geographical
- physical
- organizational
- social
- user
- task
- action
- technological
- time

As described in more detail in section 4.1, a single sensor does not seem to be appropriate to capture the different aspects of context.

## 3. Related Work

Context awareness is currently a highly active research topic [2], but most publications assume few but powerful sensors like video or infrastructure based location-tracking. Albrecht Schmidt, Hans-Werner Gellersen and Kristof van Laerhoven have presented an architecture for recognizing context from multiple simple sensors[21][34][35][36] in the TEA and Smart-Its projects. Our work takes a comparable approach to context detection by using multiple diverse sensors, but extends it to also exploit qualitative, non-numerical features [24]. Additionally, our framework introduces the prediction of future context, which has not been considered in the TEA project. The ORESTEIA project is concerned with hybrid intelligent artifacts, but depends on a priori training of artifacts by a vendor in a special training phase and explicit retraining phases for adaptation [28]; we seek to avoid the distinction of operation and training phases so that a device can be fully operational at all times.

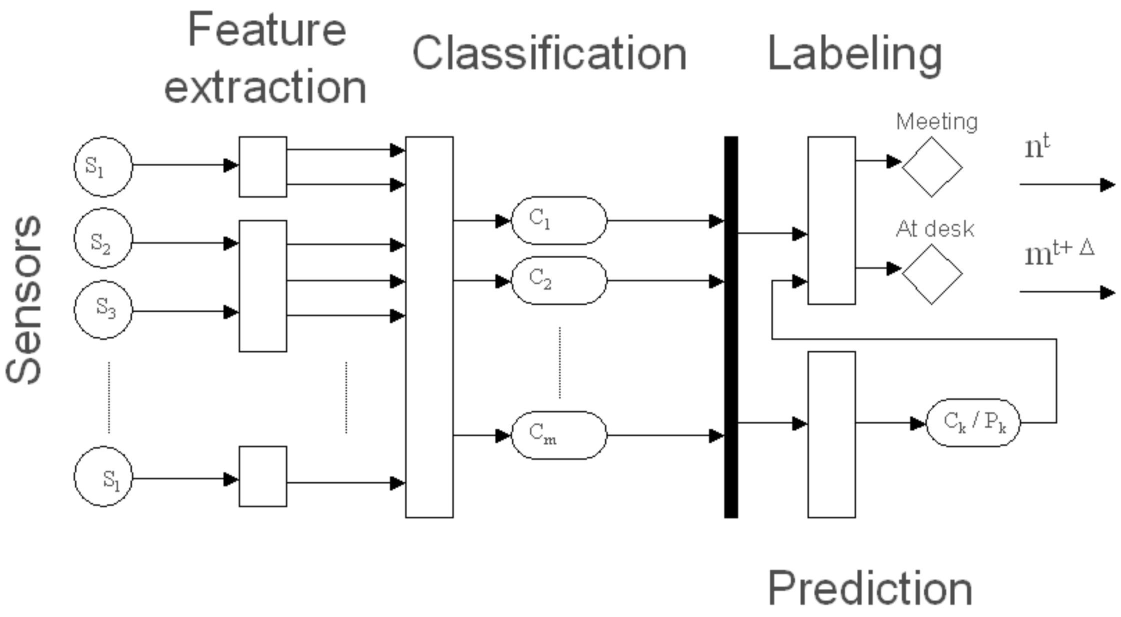
Feature extraction from different types of sensors has also been described in various publications in the field of context awareness, e.g. [4] describes the use of K-Means clustering and HMM to obtain context from a microphone, while [3] describes the use of audio and video for context detection, and others [5][17][23]. Our notion of features is equal to “cues” in the TEA project or to “Contextual Information Providers” in CIS [19]. In the field of robotics, feature extraction and sensor fusion have been studied extensively, but with a different focus. For autonomous robots, the geometrical properties of the environment (e.g. surfaces, angles, edges, color, textures, etc.) are of utter importance and need to be determined accurately to avoid potential collisions. Sensor fusion provides an appropriate means of combining multiple different sensors to resolve ambiguities, increase robustness due to redundancy and determine different properties of the same real-world objects [1]. For context awareness in information appliances, sensor fusion is at the current state of research not appropriate, because the available sensors typically capture different, mostly orthogonal aspects of the user or device context. Fusing of sensors necessitates some level of redundancy and a common model, which is currently not available for context descriptions. A possibility for exploiting multiple similar sensors, which can obviously be fused, is to exchange raw sensor or feature vectors between devices in spatial proximity. If two or more devices have similar sensors, their samples can be merged to obtain a possibly more complete view on the environment. This has been proposed independently in [23], similar to our recommendation in section 4.1.

In [7], Anind K. Dey et.al. described a software infrastructure for context awareness, which depends on a server for aggregating context and is limited to discrete-valued types of sensors. An implementation of this infrastructure is the Context Toolkit [32]. This toolkit is not directly comparable to our work; we aim to implement context recognition and prediction locally on each device, without the need for infrastructure components, while the Context Toolkit intentionally is an infrastructure approach.

Proactive adaptation of applications on the base of context has also been explored in [19]. The “Contextual Information Service” provides a lightweight interface for obtaining context information, but follows the approach of adapting the environment. We intend to autonomously adapt the device to a changing environment, which includes changing user behavior.

Learning user’s habits has previously been explored by Michael C. Mozer in The Neural Network House [25], which is able to predict occupancy of rooms, hot water usage and likelihood that a zone is entered in the next few seconds using trained feedforward neural networks. Kidd et.al. reported [20] about the Aware House, which should also learn user’s habits, but was not finished at the time of the report. The MavHome project [6] by Diana J. Cook et.al. also utilizes prediction algorithms to forecast user actions, but parts of the prediction seem to rely on database support and batch training.

Time series forecast has been explored in different areas, including distributed simulation [10], software agents [31], data value prediction in processors [33], data mining from health records [38] and theoretically for neural networks [37].



$$\langle s_1, s_2, ..., s_l \rangle_t \rightarrow \langle f_1, f_2, ..., f_n \rangle_t \rightarrow \langle c_1, c_2, ..., c_m \rangle_t \rightarrow \langle c_1, c_2, ..., c_m \rangle_{t+\Delta}$$

| Input vector<br>(Sensor vector) | Feature vector | Class vector | Future class vector |
|---|---|---|---|

Fig. 1 Architecure for proactivity via predicted user context

Utilizing different types of features for context recognition and the use of time series forecast methods for predicting future context on the level of aggregated context identifiers is, to the best of our knowledge, a new approach and has not been covered before by published research.

## 4. Architecture

Sensor readings are classified to detect common patterns in the input values. These patterns are interpreted as “states” of an abstract state machine that act as context identifiers. A user context is therefore abstracted to these states, whose internal data structures relate sensor readings to states. Although this interpretation makes it more complicated for applications to query for specific aspects of a context (e.g. location) instead of the context identifier, it allows to monitor and record the state trajectory of this abstract state machine. When a user advances from one context to another, sensor readings will change and another state will become active, reflecting the context change. Thus, interpreting the context changes as a state trajectory allows to forecast the future development of the trajectory, and therefore to predict the anticipated context. For clarity, we will only use the term context in the remainder of this article, which is similar to a state in our interpretation.

It is important to note that context classification and prediction must be performed in real-time for any practical application; it is not feasible to log data and process it offline. For the vision described in section 1, an information appliance will have to be continuously running and always be able to provide services to the user. Therefore, our architecture is targeted towards embedded systems, running without user intervention for arbitrarily long periods.
To derive knowledge about the device/user context from raw sensor data, the following steps are applied, which are depicted in Fig. 1:

1. Sensor data acquisition: Sensors, e.g. brightness, microphone or IEEE802.15 Bluetooth and IEEE802.11b Wireless LAN (*WLAN*) network interfaces, provide data streams (time series) of measurements. Usually some physical values like the incoming RF signals are the base for measurements, but more abstract sensors like the currently active application can also be utilized. In [34], sensors are classified as physical or logical, which we are not doing for a variety of reasons. Within our work, a sensor is any entity that can provide measurements of the environment a device is situated in. Values are sampled at discrete time steps $t \in \mathrm{N}_0$, whose frequency should be set to the maximum desired sample frequency of the used sensors. As this is highly application specific, no general distance between sample time steps can be determined.

A *sensor vector* $S_1 \times S_2 \times \ldots \times S_l$ defines sensor readings $\langle s_1, s_2, \ldots, s_l \rangle_t \in S_1 \times S_2 \times \ldots \times S_l$ for points in time $t$.

2. Feature Extraction: From raw sensor data, information can be extracted by domain-specific methods, yielding multiple data values per sensor, which are called *features* $F$ with samples $f \in F$ as a function of time. During feature extraction, the available data is deliberately simplified, transformed or even expanded, allowing it to be interpreted better. Usually, simple statistical parameters like the mean $\bar{x}$, standard deviation $s$ or higher moments are used as features for time series of numerical data. For nominal and ordinal data, alternative methods should be explored.

   The set of features is called a *feature vector* $F_1 \times F_2 \times \ldots \times F_n$, which defines samples $\langle f_1, f_2, \ldots, f_n \rangle_t \in F_1 \times F_2 \times \ldots \times F_n$ for points in time $t$ in the multi-dimensional *feature space*.

   Although this definition does not allow for meta attributes on feature values, it might be advantageous to do so. In the CIS project [19], meta information like confidence or accuracy is added to feature values and can be used by applications. In the architecture presented in this article, a confidence value might be mapped to a weight in the classification step to adaptively weaken the influence of features with (currently) poor sampling quality.

3. Classification: The task of classification is to find common patterns in the feature space, which are called *classes* or *clusters*. Because a feature vector should possibly be assigned to multiple classes with certain *degrees of membership* (the "probability" or "confidence" that the feature vector belongs to a class), soft classification / soft clustering approaches are utilized. These approaches map a feature vector of $n$ different features to degrees of membership $c \in C$ with $C := [0;1]$ of $m$ different classes: $F_1 \times F_2 \times \ldots \times F_n \rightarrow C^m$. The classes $c_i$ for $i = 1 \ldots m$ are regarded as the detected user context and are identified by a simple index in the class vector.

   The *class vector* $C^m$ defines class degrees of membership $\langle c_1, c_2, \ldots, c_m \rangle_t \in C^m$ for points in time $t$.

4. Labeling: To ease the development of context-aware applications and for presenting detected context to the user, descriptive names should be assigned to single classes or combinations of classes (cf. [21]). Labeling maps class vectors to a set of names: $C^m \rightarrow N$ where $N$ is a set of user-defined context names (strings).

   A *context name* or *context label* $n_t \in N$ describes the currently active context for points in time $t$.

5. Prediction: To enable proactivity, our approach is to forecast future context. Thus, the prediction step generates anticipated future class vectors from current ones: $C^m \times R^+ \times R^+ \rightarrow C^m$.

   A (future) class vector defines degrees of membership for each class: $\langle c_1, c_2, \ldots, c_m \rangle_s = p(\langle c_1, c_2, \ldots, c_m \rangle_t, t, s)$ for points in time $t$ and $s$ with $s > t$.

The following sections describe these five blocks in more detail.

## 4.1 Sensors

Context awareness of information appliances premises that they can rely on context-relevant information gathered by sensors. The acquired information should be as close to the user's world perception as possible [34]. Unlike sensing information in other domains (e.g. quality assurance, robotics, etc.), where the object of interest is explicitly investigated for the sake of accurate and highly reliable measurement reading, sensors for Pervasive Computing information appliances have to be less intrusive and ostensible. Furthermore, for collecting context information, varieties and events in the measured data are much more interesting than the actual sensor output; thus different techniques and methods are required [9]. Gellersen et.al. proposed the use of diverse simple sensors as an alternative to the integration of a single generic sensor. Presuming that current information appliances are already equipped with sensors that can be exploited for those means, this approach is more rational. The variety of different sensor types results in a better representation of the users context than a single generic sensor [13]. Examples of such sensors in a typical information appliance are listed in Table 1, while Table 2 lists additional sensors that might be useful for recognizing user context and can be easily added to current and future information appliances.

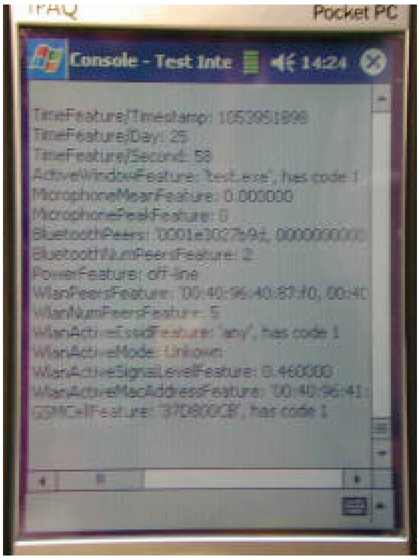
Table 1 Typical sensors available in a mobile device

| |
|---|
| time |
| microphone |
| brightness |
| Bluetooth |
| Wireless LAN |
| (un)docked |
| logged on(pff |
| application manager |

Table 2 Additional sensors for a mobile device

| |
|---|
| GPS |
| GSM |
| compass |
| accelerometer |
| tilt sensor |
| temperature sensor |
| pressure sensor |
| various bio-medical sensors |

Fig. 2 Feature extraction on a typical PDA with a mobile phone as additional sensor



```
TimeFeature/Timestamp: 1053951898
TimeFeature/Day: 25
TimeFeature/Second: 58
ActiveWindowFeature: 'test.exe', has code 1
MicrophoneMeanFeature: 0.000000
MicrophonePeakFeature: 0
BluetoothPeers: '0001e3027b9d, 000000000
BluetoothNumPeersFeature: 2
PowerFeature: off-line
WlanPeersFeature: '00:40:96:40:87:f0, 00:40
WlanNumPeersFeature: 5
WlanActiveEssidFeature: 'any', has code 1
WlanActiveMode: Unkown
WlanActiveSignalLevelFeature: 0.460000
WlanActiveMacAddressFeature: '00:40:96:41:
GSMCellFeature: '37D800CB', has code 1
```

Fig. 3 Feature extraction on a typical PDA with a mobile phone as additional sensor: example values

To improve the quality of context recognition, information appliances can share their perceptions to create a more complete model of the current context. This is accomplished by mutually granting access to the raw sensor data of devices in the neighborhood in a peer-to-peer manner. Own sensor data can be correlated with the data of sensors in range, increasing the accuracy. Nevertheless, context information can only be shared within a close range to ensure that the recognized context is still local and distinct and not a global representation of different neighboring situations. By this means, the list of sensors is easily extensible by equipping the user with smart sensors that expose their information to a close, interested (and authorized) device. E.g., a biological sensor could measure the user's pulse and transmit it to the information appliance via Bluetooth or similar ad-hoc communication methods. Fig. 2 shows an example of using a mobile phone for retrieving GSM sensor data via Bluetooth. The list of processed sensor information is only limited by processing capabilities and memory of the information appliance.

## 4.2 Feature Extraction

Although feature extraction and classification are well-known fields of research, most publications only cover numerical, continuous features. Recently we introduced a model for utilizing heterogeneous features (e.g. a list of Bluetooth or WLAN devices in range in combination with the time stamp) in a common classification step [24].

The feature vector $\left\langle f_1, f_2, \ldots, f_n \right\rangle$ formed by an arbitrary combination of these features is highly heterogeneous; therefore, it is necessary to find a way to cope with the different types and semantics of the feature space dimensions in the classification step. In our concept, a feature type is defined by the feature extractor that does the actual transformation of raw sensor data into the more relevant exposition of the data. Therefore, these transformations can be done independently for each feature and are completely domain specific; each feature type can implement its operators needed for classification differently. This abstraction virtually maps different kinds of sensor data and their respective feature types to a unified multidimensional, homogeneous feature space that can be classified by commonly used algorithms.

Feature types can be categorized as one of the following (a similar, albeit not identical taxonomy has been used in [26]):

- nominal (categorical, qualitative): The feature takes on values of a set on which no order relation has been or can be defined. A special case are binary features with $F = \{0,1\}$.

- ordinal (rank): The feature takes on a values of a set with a defined order relation $<: F \times F$.
- numerical (quantitative): The feature takes on values of an ordered set with defined $+$ and $\bullet$ operations (an algebraic field). It can be further distinguished according to the density of values in the set:
  - discrete: $F \in Z$
  - continuous: $F \in R$
- interval: The feature takes on intervals instead of single values, e.g. $F \subseteq Pot(R)$.

A preliminary comparison of different classification methods, ranging from clustering algorithms to neural networks, showed that only two operations are necessary on an abstract feature $F$: a distance metric and an adaptation operator [24]. With these two operations, supervised and unsupervised classifiers like the Kohonen Self-Organizing Map (*SOM*), K-Means clustering or Lifelong GNG [16] can be easily applied to any feature which defines them. In Fig. 3, an example list of features in a typical mobile scenario is shown on the PDA screen, including lists of Bluetooth and WLAN devices in range, the current GSM cell (queried from a mobile phone via Bluetooth) and specific audio features from the microphone. Each of these features has been implemented with appropriate distance and modification operators.

## 4.3 Classification

The classification step is used to find similarities in and learn recurring patterns from its input data. It serves the input for the labeling and the prediction steps in form of a probability vector containing the probability of activity for each learned class.

A classification algorithm has to fulfill several requirements to be applicable for classifying sensor data and recognizing context:

- On-line learning: There is no dedicated training phase, learning has to be done unsupervised and continuously.
- Adaptivity: Learning must never stop; as user's habits will change over time, classes must always adapt to new input data. This prevents the use of a continuously decreasing learning rate as used in many methods (e.g. some neural networks).
- Variable topology: Because the number of classes can not be determined a priori for the general case, the internal topology must be able to adapt dynamically.
- Soft classification: Context classes are not mutually exclusive, more than one context can be active at the same time (e.g. 'at home' and 'sleeping').
- Noise resistance: When working with real-world data, the algorithms have to cope with the intrinsic noise that is sampled with all signals.
- Limited resources: The algorithm has to work within the capability constraints of an information appliance (small RAM, little processing power, etc.).
- Simplicity: In our case, the algorithm should perform as few distinct operations on the feature vector as possible. As the feature extractors have to provide the necessary operators (see feature extraction), a multitude of operators drastically complicates the implementation.

Ideally, a classification algorithm must be on-line and thus unsupervised and must have a variable network topology to cope with changing feature vector dimensionality (changing sensor configurations). The classification algorithm must not be hard competitive to allow multiple active contexts and it has to be designed for life long learning to not forget or overwrite already learned clusters over time (which is known as the plasticity-stability dilemma [15] in neural networks and clustering literature). Table 3 shows a comparison of the most common on-line clustering algorithms and serves as a base for selecting the most appropriate one. K-Means, Leader, G K-Means and IDBSCAN segregate themselves due to their hard competitive classification strategies. SOM and RSOM tend to forget their previously learned classes very quickly due to their learning strategy and fixed network topology. Although this can be circumvented by combining the SOM with K-Means clustering [21], GNG [11] still seems to provide more flexibility in environments with changing configurations.

In [16], Hamker proposed modifications to the original GNG algorithm to cope with continuously changing environments and life-long learning (LLGNG). These modifications prevent the GNG from growing permanently by introducing a learning rate with a locality criteria. This results in a locally converging but globally still adaptive learning algorithm. New classes will always be learned but changes in already learned classes are only applied if the cluster representing this class does not match the new input vector properties to a reasonable extent. Due to these modifications, the algorithm also performs better in environments with small memory because a new cluster always represents a new context and is never redundant. Therefore, LLGNG can forget the oldest and most erroneous cluster when the memory boundaries are reached; this ensures that memory is always available for learning new classes. The basic rule behind learning and insertion in the LLGNG is that *"organisms only learn when events violate their expectations"*, previously assumed by [30]. Tests and performance evaluations are in work and will be presented in more detail in future publications.

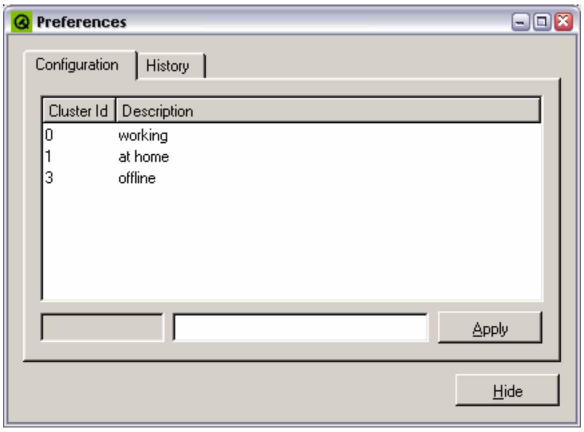
Table 3 General overview of algorithms for unsupervised clustering of sensor data, based on [21]: [1]

| ***On-line Algorithm*** | ***Network Topology*** | ***Topology Preserving*** | ***Competitive*** |
|---|---|---|---|
| SOM | fixed | yes | soft |
| RSOM | fixed | yes | soft |
| K-Means | fixed | no | hard |
| Leader | variable | no | hard |
| G K-Means | variable | no | hard |
| Neural Gas | variable | no | soft |
| NG+CHL | variable | yes | soft |
| GNG | variable | yes | soft |
| IDBSCAN | variable | no | hard |

## 4.4 Labeling

Applications will generally be unaware of classes and their current degrees of membership. In real world scenarios, it would be virtually impossible to design applications to work with these class vectors, because they are learned in an unsupervised way and will therefore differ from one device to another; class vectors depend on the order in which those classes were detected first. Therefore, the indices of the currently active classes need to be mapped to more meaningful values. In our architecture, simple strings are used as context labels allowing users to enter them. This is the only step where user interaction is necessary and even in this case, it can be non-obtrusive. Approaches for such an interface include a discreet icon in one corner of the device screen which blinks when a still unlabeled context class has been active for some period of time, allowing a user to assign a name for the current context. Another option is to display an automatically created context log in form of a diary, which allows the user to label formerly detected context classes.

The complexity necessary for this step mostly depends on the quality of the classification step. If classes are long-term stable, i.e. previously learned classed are not overwritten by different new ones, then a simple 1-to-1 mapping of classes to labels might be enough. However, if the used classification algorithm overwrites older classes in order to learn new context, then the degrees of membership of all classes will need to mapped to labels. In [21], a simple K-Means clustering algorithm is used as a second step after clustering. For each class, represented by a winner neuron of the Kohonen SOM clustering, K-Means is applied to the input vectors of the SOM (which correspond to feature vectors in our architecture) to avoid overwriting of labels. This added complexity is necessary because of the shortcomings of the SOM, and one of the reasons for selecting GNG for our first experiments.



| Cluster Id | Description |
|---|---|
| 0 | working |
| 1 | at home |
| 3 | offline |

Fig. 4 GUI prototype: Allowing the user to assign descriptive labels

Fig. 5 GUI prototype: non-obtrusive display of current context

The first implementation is based on a direct 1-to-1 mapping and provides the user a non-obtrusive way to assign a label to a class. The prototype labelling application is currently realized independently from the framework to permit autonomous development and testing. The communication between the labelling application and the classification framework occurs via a simple SOAP protocol (a DCOM interface on windows is also available), not being restricted to a single platform. Fig. 4 shows the dialog used to assign labels to the classes found by the classification algorithm, Fig. 5 shows how the application signals the user the currently active context. It is also

---

[1] Kohonen Self-Organizing Map (*SOM*), the Recurrent Self-Organizing Map (*RSOM*), K-Means clustering, Hartigan's sequential leader clustering, growing K-means clustering, neural gas, neural gas with competitive Hebbian learning (*NG+CHL*), growing neural gas (*GNG*) and incremental DBSCAN (*IDBSCAN*). Unlike Van Laerhoven we rate NG+CHL and GNG as topology preserving [12].

possible to view the context history to assist the labelling of new contexts.

It is still an open issue if the classification quality will facilitate the use of a simple 1-to-1 mapping or if a more complex n-to-1 mapping from the whole class vector to labels will perform better. When an appropriate solution has been found for a wide range of application areas, the labeling will be incorporated directly into the framework. Thus, the labeling application will only be responsible for the naming of recognized contexts, but not for storing the associations or the context history.

## 4.5 Prediction

Prediction is the main novelty in our architecture and the focus of our current research. As prediction should again be performed without user interaction, it is not necessary to work with labeled context. Instead, the prediction step in our architecture builds upon the class vector generated by the classification step. This allows to predict more than a single future "best matching" context by exploiting the class degrees of membership (which would be impossible if the prediction would take the single "best matching", labeled context as input).

The aim is to generate class vectors for future points in time, which have the same meaning as the current class vector provided by the classification step. This allows to feed the predicted class vectors into the labeling step to provide predicted context labels for use in proactive applications (cf. Fig. 1). Before going into more detail, it is generally good to first analyze the requirements for a prediction algorithm in this sense.

- Unsupervised model estimation: Model topology and parameters need to be estimated automatically without user interaction or explicit definition of input/output behavior.

- On-line learning: For embedded devices in real-world scenarios, it is infeasible to switch between artificially separated training and prediction phases or even to store enough history for a batch training. Therefore, the algorithm has to continuously adapt its parameters during normal operation, incorporating new class vectors as soon as they arrive.

- An exception to this is to store only the recent history in detail, which could be used to optimize

- and/or evaluate the quality of the predictions (by splitting the history in a training and a test set).

- Incremental model growing: When new classes are detected in the classification step during run-time, new dimensions will be added to class vectors. The prediction algorithm must be able to incrementally increase its internal model topology without requiring a complete retraining, e.g. by initializing new dimensions with default values. It is currently unclear if shrinking of class vectors during run-time is also necessary or if "dead classes" could simply receive a minimum probability.

- Confidence estimation: The algorithm should be able to compute an estimation of the correctness of the forecasted context along with the forecast itself. This estimation can be used by the application as a confidence measure to determine if the prediction should be relied on for certain actions.

- Automatic feedback: The prediction engine should continuously estimate the next class vectors and evaluate its estimations by comparing with the real class vectors when they are available.

- Manual feedback: If some action that has been carried out automatically due to a forecast is reverted/canceled by the user, this forecast should receive a penalty to make it less likely the next time (this is known as reinforcement learning in machine learning) .

- Long-term vs. short-term: The used method should ideally be suitable (e.g. parameterizable) for different forecasting horizons, i.e. predicting context in the near future with high confidence, but also being able to predict later context, most probably with lower confidence.

We have currently not selected a specific algorithm for the prediction step because our architecture is open for arbitrary algorithms that can be adapted to suit our interface. However, after first research on possible candidates, Markov predictors seem to be generally suited well. Active LeZi [14] has already been implemented as predictor plug-in within the framework and is currently used for initial experiments on real-world data. Although prediction accuracy on simple artificial data sets looks promising, the results with real-world data suggest further research. Another special form of Markov predictors, the Variable Duration Hidden Markov Models (VDHMM) also seem to be applicable for predicting context trajectories. They explicitly model the duration distributions and are thus capable of predicting for different forecasting horizons and, as for nearly all variants of HMMs, there exist mature methods for learning model parameters. It might be necessary to use multiple different predictor plug-ins concurrently and fuse their results to generate a

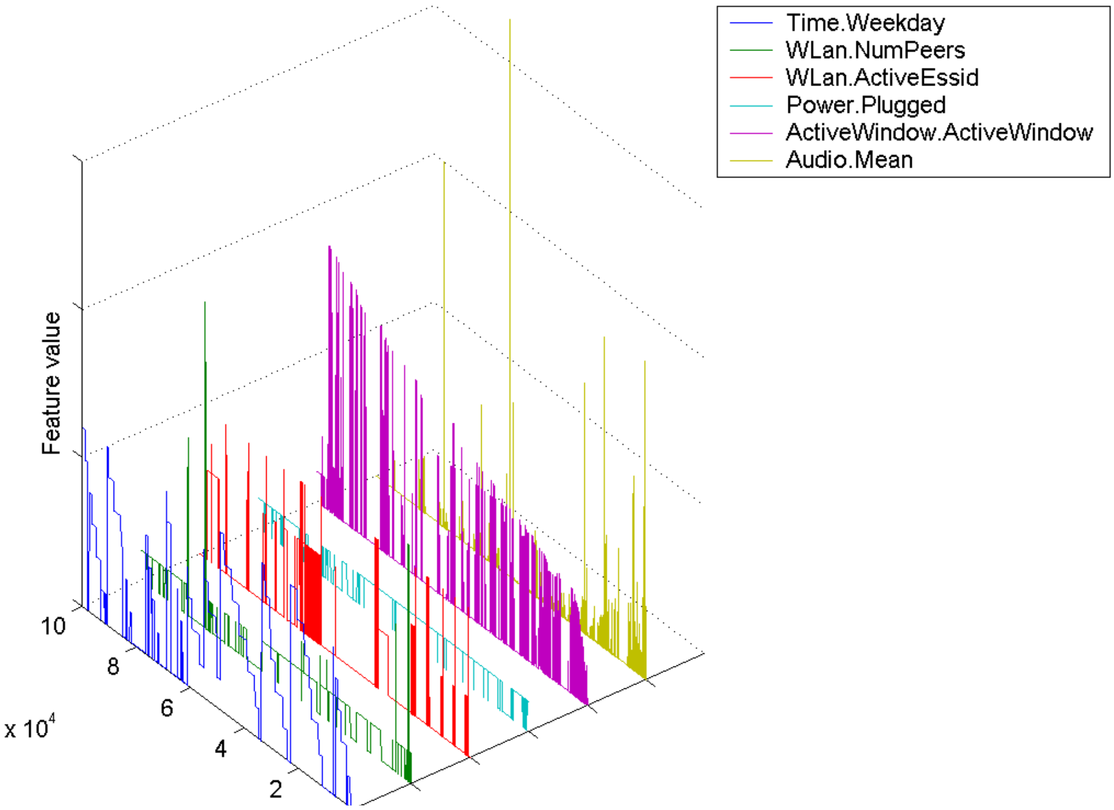


Fig. 6 Feature value trajectories of the complete data set

reasonable forecast of the context trajectory. A single prediction method is usually unable to respect trends and periodic patterns in the context history as well as performing sequential Prediction by Partial Match (*PPM*). For combining the results of different predictors, it is important that a confidence estimation is generated by the method.

## 5. Implementation

The described architecture has been implemented in form of a cross-platform framework which is freely available under an open source license. Interfaces provided by the framework are implemented by adhering to standardised protocols like SOAP and DCOM to guarantee most flexibility when developing context aware third party applications.

Currently it runs under Windows 2000 or XP, Linux on IA32 and ARM processor platforms, Windows CE 3.0 and, with restrictions, under Symbian OS 7.0.

## 6. Experiments and Evaluation

We have evaluated our framework for context recognition and prediction on two real-world data sets. The first data set has been gathered over 3 weeks on one of the servers for our smart room and includes the list and number of Bluetooth devices in the room and the list and number of Wireless LAN clients in the room. For the second data set, a broader range of sensors has been used. On a standard notebook computer, which is used for daily work, the following features were recorded over a period of about 2 months with over 100000 samples: weekday, active window (i.e. active application), mean environmental loudness, plugged into charger, WLAN ESSID, WLAN mode, WLAN signal level, WLAN access point MAC address, Bluetooth peers in range, number of Bluetooth peers in range and the GSM cell ID of the mobile phone (which was connected via Bluetooth). Fig. 6 shows the trajectory of 6 out of these 11 feature values over the whole recording period – the other features are not shown because they either yield non-atomic values or do not contribute significantly to context recognition.

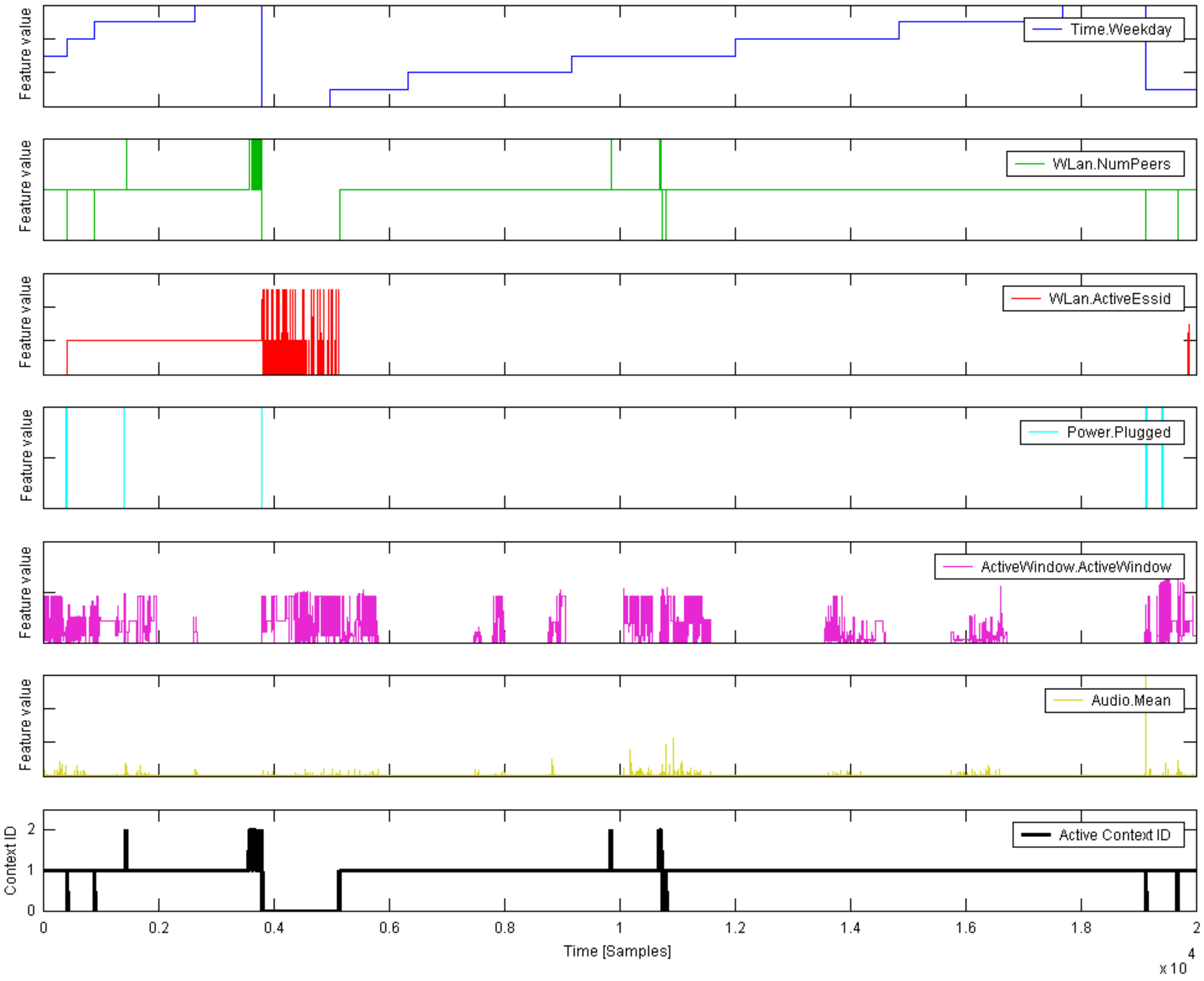


Fig. 7 Recognized context identifiers

In Fig. 7, a part of the data set over roughly 2 weeks is shown in more detail. The first 6 plots depict the feature values, while the last one shows the respective context ID with highest activation after the classification step. As can be seen, the classification algorithm selected 3 different contexts during this time frame.

## 7. Conclusions and Future Work

We have presented an architecture to recognize and predict user context by utilizing multiple heterogeneous sensors. This architecture consists of four steps: feature extraction (to generate a more relevant representation of sensor data, exploiting domain-specific knowledge), classification (to find similarities and common patterns in the input data), labeling (to assign simple context names to recognized classes) and prediction (to forecast future user context based on past behaviors). The novelties in this approach are the prediction of possible user actions via context forecast and the abstraction of feature types to allow heterogeneous features to be combined in a single classification step. To accomplish this, all feature types independently define the operations necessary for classification.

We have already implemented feature extraction for various sensors available on typical information appliances, including microphone, Bluetooth, Wireless LAN and additional, external sensors like a mobile phone accessible via Bluetooth. A next step in research will include gathering real-world data in an empirical study and evaluating classification and prediction algorithms based on this data.

Proactivity in applications can support users by allowing information appliances to adapt to the user instead of forcing the user to learn specifics of the interface. When equipped with multiple sensors and using those sensors to detect and predict context, an information appliance can become smarter and more intuitive to use, fostering a wider acceptance of information appliances in everyday life.

## Acknowledgments

We especially thank Manfred Hechinger and Günther Blaschek for helpful discussions on the general topic of proactivity and the implications on user interfaces.

### References

[1] S.B.H. Bruder. An information centric approach to heterogeneous multi-sensor integration for robotic applications. *Robotics and Autonomous Systems*, 26(4), March 1999.

[2] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[3] B. Clarkson, K. Mase, and A. Pentland. Recognizing user context via wearable sensors. In *ISWC*, pages 69–76, 2000.

[4] B. Clarkson, N. Sawhney, and A. Pentland. Auditory context awareness via wearable computing. In *Proceedings of the 1998 Workshop on Perceptual User Interfaces (PUI'98)*, San Francisco, CA, USA, November 1998.

[5] B. Clarkson and A. Pentland. Unsupervised clustering of ambulatory audio and video. Technical Report 471, MIT Media Lab, Perceptual Computing Group, 1998.

[6] D. J. Cook, M. Youngblood III, E. O. Heiermann, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. MavHome: An agent-based smart home. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pages 521–524, IEEE Computer Society Press, March 2003.

[7] A. Dey, G. D. Abowd, and D. Salber. A context-based infrastructure for smart environments, 1999.

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

[9] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, January–March 2002.

[10] A. Ferscha. Adaptive time warp simulation of timed petri nets. *IEEE Transactions on Software Engineering*, 25(2):237–257, March/April 1999.

[11] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

[12] B. Fritzke. Some competitive learning methods. Technical report, Systems Biophysics, Inst. for Neural Comp., Ruhr-Universität Bochum, April 1997.

[13] H.W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artefacts. *Mobile Networks and Applications*, 7(5):341-351. ACM Press, October 2002.

[14] K. Gopalratnam and D. J. Cook. Active LeZi: An incremental parsing algorithm for sequential prediction. *Proceedings of the Florida Artificial Intelligence Research Symposium*, 2003.

[15] S. Grossberg. Adaptive pattern classification and universal recoding: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[16] F. H. Hamker. Life-long learning cell structures—continuously learning without catastrophic interference. *Neural Networks*, 14(4–5):551–573, May 2001.

[17] R. Headon. Movement awareness for a sentient environment. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pages 99-106, IEEE Computer Society Press, March 2003.

[18] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of Conference on Human Factors in Computing Systems (CHI '97)*, pages 234–241. ACM, March 1997.

[19] G. Judd and P. Steenkiste. Providing contextual information to pervasive computing applications. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pages 133-142, IEEE Computer Society Press, March 2003.

[20] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner, and W. Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Cooperative Buildings, Integrating Information, Organization, and Architecture, Second International Workshop, CoBuild'99*, volume 1670 of *Lecture Notes in Computer Science*, pages 191–198. Springer, 1999.

[21] K. Van Laerhoven and S. Lowette. Real-time analysis of data from many sensors with neural networks. In *Proceedings of the fourth International Symposium on Wearable Computers (ISWC) Zurich, 7-9 October 2001*. IEEE Press, 2001.

[22] S. Mann. Wearable computing: A first step toward personal imaging. *IEEE Computer*, 30(2), February 1997.

[23] J. Mäntyjärvi, J. Himberg, and P. Huuskonen. Collaborative context recognition for handheld devices. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pages 161-168, IEEE Computer Society Press, March 2003.

[24] R. Mayrhofer, H. Radi, and A. Ferscha. Feature extraction in wireless personal and local area networks. In *Proceedings of The Fifth IFIP TC6 International Conference on Mobile and Wireless Communications Network (MWCN 2003)*. World Scientific, October 2003.

[25] M. C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*, pages 110–114. AAAI Press, 1998.

[26] S. Negri and L. Belanche. Heterogeneous kohonen networks. In José Mira and Alberto Prieto, editors, *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks (IWANN 2001)*, Lecture Notes in Computer Science, pages 243–252. Springer, 2001.

[27] D. A. Norman. *The invisible computer*. MIT Press, Cambridge, 1998.

[28] Deliverable D05: 1st year progress report of the Oresteia project. Technical report, January 2002.

[29] F. Pichler. *Mathematische Systemtheorie: Dynamische Konstruktionen*. Walter de Gruyter, Berlin, New York, 1975.

[30] R. A. Rescorla and A. R. Wagner. A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and non-reinforcement. In A. H. Black and W. F. Prokasy, editors, *Classical Conditioning II. Current Research and Theory*. Appleton Century Crofts, New York, 1972.

[31] M. T. Rosenstein and P. R. Cohen. Concepts from time series. In *AAAI/IAAI*, pages 739–745, 1998.

[32] D. Salber, A. K. Dey, and G. D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI'99)*, pages 434–441, May 1999.

[33] Y. Sazeides and J. E. Smith. The predictability of data values. In *International Symposium on Microarchitecture*, pages 248–258, 1997.

[34] A. Schmidt. *Ubiquitous Computing – Computing in Context*. PhD thesis, Lancaster University, November 2002.

[35] A. Schmidt and M. Beigl. There is more to context than location: Environment sensing technologies for adaptive mobile user interfaces. In *Workshop on Interactive Applications of Mobile Computing IMC'98*, 1998.

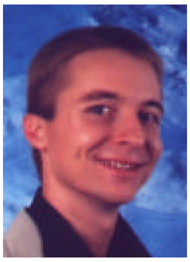
[36] A. Schmidt and K. Van Laerhoven. How to build smart appliances. *IEEE Personal Communications*, August 2001:66–71, 2001.

[37] Z. Tang and P. A. Fishwick. Feed-forward neural nets as models for time series forecasting. Technical Report 91-008, University of Florida, 1993. also published in ORSA Journal of Computing 5(4), 374–386.

[38] M. A. Orgun, W. Lin and G. J. Williams. Multilevels hidden markov models for temporal data mining. In *Proceedings of the KDD 2001 Workshop on Temporal Data Mining (held in conjunction with the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001))*, San Francisco, CA, USA, August 2001.

[39] M. Weiser. The computer of the twenty-first century. *Scientific American*, 1496:94–100, September 1991.

[40] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. HTTP://www.doc.mmu.ac.uk/STAFF/mike/ker95/ker95-html.h (Hypertext version of Knowledge Engineering Review paper), 1994.



**Rene Mayrhofer** received the Dipl.-Ing. degree in Computer Science from the Johannes Kepler Universität Linz, Austria in 2002 and is currently working on his PhD on context prediction, also at Johannes Kepler Universität Linz. His research interests include context awareness, embedded systems and artificial intelligence. He is a contributing developer of the Debian GNU/Linux project.

**Harald Radi** is currently working on his Dipl-Ing. degree in Computer Science at Johannes Kepler Universität Linz, Austria. His research interests include component technologies, artificial intelligence and context awareness. He is a member of the PHP development team and contributor to the PEAR repository.

**Alois Ferscha** received his Mag. degree in 1984, and a PhD degree in business informatics in 1990, both from the University of Vienna, Austria. From 1986 through 2000 he was with the Department of Applied Computer Science at the University of Vienna at the levels of assistant and associate professor. In 2000 he joined the University of Linz as full professor where he is now head of the Institute for Pervasive Computing. He has served on the committees of several conferences like Pervasive Computing, UMBICOMP, WWW, PADS, DIS-RT, SIGMETRICS, MASCOTS, TOOLS, PNPM, ICS, etc. and is currently the program chair for the PERVASIVE 2004 conference.