# PREDICTION OF CONTEXT TIME SERIES

Stephan Sigg
*Chair for communication Technology*
*University of Kassel*
*D-34121 Kassel, Germany*
*sigg@uni-kassel.de*

Sandra Haseloff
*Chair for communication Technology*
*University of Kassel*
*D-34121 Kassel, Germany*
*sigg@uni-kassel.de*

Klaus David
*Chair for communication Technology*
*University of Kassel*
*D-34121 Kassel, Germany*
*sigg@uni-kassel.de*

**ABSTRACT**

Context awareness is a key feature of modern computing applications, allowing application behaviour to be adapted to context. The power and quality of context-aware applications can significantly be increased by not only considering past and present contexts for adaptation, but by also predicting and reacting to future contexts. This paper deals with fundamental challenges of context prediction. After an introduction to context prediction and a clarification of the basic terms associated with it, the paper proposes solutions concerning prediction algorithms and describes an architecture for context-aware computing, including context prediction, we have developed. The findings described in the paper are illustrated and verified by simulations using different types of context data and different prediction algorithms.

**KEYWORDS**

Context awareness, Context prediction.

## 1. INTRODUCTION

If we say that a device is context aware we do not explicitly state if it is aware of his present, past or future context [1]. Most work is done on considering present or past context. However, some authors also consider the future context. The latter case of context computing is usually referred to as context prediction, forecasting or proactivity. While the term context prediction is most prominently used in conjunction with context awareness [1], proactivity was originally considered for software agents [2]. The term forecasting is most often found in relation to stochastic time series analysis [3, 4, 5]. Most prominent application fields are the financial or stock market [6, 7].
However, the different notions become mixed as some authors also use the terms of forecasting or proactivity in order to describe the context prediction process [8, 9]. Some authors even use these notions in order to describe the process of inferring a context [10, 11]. Furthermore, the use of the term context prediction is not uniform among researchers. While the authors in [1, 9, 12] use the term context prediction to describe an operation that infers future contexts from past and present contexts, [10] uses this term in order to

describe the automatic triggering of actions when some context becomes active, while the authors of [13, 14] use it to describe the process of inferring context from sensor outputs.
In our understanding, context prediction can be used by applications to react to events that will likely occur in the future. The information base on the user context is therefore expanded by context prediction.
The cost for this additional information is an increased error probability of the predicted context information. It lies in the nature of prediction that the reliability of a predicted element is typically not comparable to observed present or past events. While the impact of weak reliability may differ from application to application, this is definitely the most serious drawback to context prediction.
The choice of the prediction algorithm is therefore serious. Popular algorithmic approaches are Markov approaches [1, 15] or statistical methods as the autoregressive, moving average or ARMA algorithms [16, 17, 18]. An alternative approach to context prediction is to utilise alignment methods in order to predict future contexts.
Prediction algorithms are intertwined with other components in context prediction architectures. Without a general architecture, the uses of context prediction are restricted to single applications. An architecture provides a standard interface that can be utilised by several applications. Since context prediction is especially useful in mobile and ubiquitous environments, the architecture should support a distribution of components as well as discovery of services and components and dynamic adding and removal of components. With Foxtrot we introduce an approach that serves this need.
The remainder of the paper is orgenaised as follows. Section 2 introduces basic concepts in context prediction that are utilised throughout our work. Section 3 proposes an architecture for context awareness that includes modules to solve context prediction tasks. In section 4 we utilise various prediction algorithms in several simulation scenarios in order to obtain simulation results that back our discussion in previous chapters. Section 5 summarises our results.

## 2. FOUNDATIONS OF CONTEXT PREDICTION

In the following sections we introduce basic concepts that are related to context prediction.

### 2.1 Concrete Context Time Series

Context prediction algorithms predict sequences of future patterns. This is done with knowledge of the sequence of observed user contexts. We refer to these sequences as context time series.

**Definition 1**

*Let $i,j,k \in N$ and $t_i$ describe any point in time. A time series T is a non-empty, ordered set of context elements $v_i$ that are attached a timestamp $t_i$. We write $T_{t_j,t_k}$ in order to express that the attached timestamps of the context elements in $T_{t_j,t_k}$ are in $[t_j, t_k]$.*

The context elements in a time series are further grouped to time series elements.

**Definition 2**

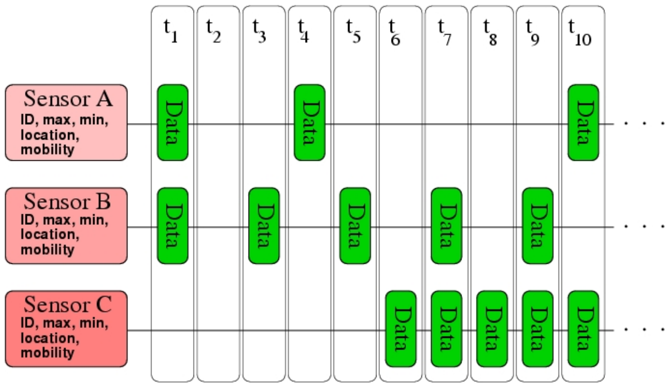
*Let T be a time series and $t_i \in R$ be a timestamp of any one context element $v_i \in T$. A time series element $\xi_i \in T$ consists of all context elements $v_i \in T$ that share the same timestamp $t_i$ ($\xi_i = v_i$ and Timestamp($v_i$)=$t_i$).*

Another property typically found in time series recorded in realistic scenarios is that time series elements contain only part of the context information at any point in time. Since different sensors most probably generate an output value at different points in time, a time series in typical realistic cases is no generic construct.

**Definition 3**

*A concrete time series T is a time series where any time series element* $\xi_i \in T$ *may contain one or more context elements of any combination of context sources.*

With concrete time series (cf. Fig. 1), operations on these time series that are otherwise straight forward become more complicated. Assume for example that we want to compare a second concrete time series with the first one in order to find similar context patterns. In most cases no subsequence of sufficient length can be found where the aligned entries are constructed from the same sensor outputs considering number and type of the sensors. A possible solution is the interpolation and extrapolation of the missing information. However, this usually increases the noise (errors) in the input data.



$t_1$ $t_2$ $t_3$ $t_4$ $t_5$ $t_6$ $t_7$ $t_8$ $t_9$ $t_{10}$
Sensor A
ID, max, min, location, mobility
Sensor B
ID, max, min, location, mobility
Sensor C
ID, max, min, location, mobility
Data
. . .

Figure 1: Illustration of a concrete time series.

## 2.2 High-level and Low-level Contexts

Context has several levels of abstraction that are commonly described by the notions high-level context, low-level context and raw sensor data [19, 20, 21]. Although a commonly accepted definition does not exist, higher context representations are typically symbolic while lower representations are numeric.

We classify the context abstraction level by the amount of processing applied to the data. The output of a sensor is considered as raw data since it most probably needs further interpretation. Sensor outputs are differing even for sensors of the same class. This is because of distinct representations or accuracies of the sensed information. A pre-processing of raw sensor data is necessary so that further processing is not influenced by special properties of the sensor itself. We refer to this pre-processing as the context acquisition step. Raw data is normalised to a representation utilised by all further context processing steps.

The low-level contexts of two sensors of the same class measured at the same time in the same place are identical with the exception of a possibly differing measurement accuracy, provided that both sensors are in good order.

In order to obtain high-level context information, the low-level contexts are further processed in the context interpretation step. Possible operations are aggregation with other low-level contexts, interpretation, data calibration, noise removal or reforming of data distributions.

There is no limit to the level of abstraction. High-level contexts may be further processed to again receive high-level context information.
In table 1 exemplary raw data, low-level contexts and high-level contexts are depicted for the keyboard sensor. The same key pressed on an English and a Russian keyboard (raw data identical) might result in different low-level contexts due to an alternative language setting (acquisition procedure). In the Cyrillic layout the letter ’ы’ is obtained while it is the letter ’z’ for the English layout.

However, for German keyboards the letters ’y’ and ’z’ are exchanged compared to the English layout, hence leading to the same low-level context even though the raw data is different. Furthermore, different context interpretation procedures may lead to distinct high-level contexts (office occupied or writing).

| High-level context | Low-level context | Raw data | Sensor |
|---|---|---|---|
| writing | z | 0x79 | keyboard |
| writing | ы | 0x79 | keyboard |
| writing | z | 0x7a | keyboard |
| office occupied | z | 0x7a | keyboard |

Table 1: Examples of high-level context, low-level context and raw sensor data for an exemplary sensor.

## 2.3 Context Prediction

The task of a prediction algorithm is to find a context sequence in an UbiComp environment that at a given point in time most likely describes the continuation of the observed time series in the future.

**Definition 4**

*Let k,m,i$\in$N and* $t_i$ *describe any point in time. Furthermore, let T be a concrete context time series. Context prediction is the task of learning a prediction function* $f_{t_i} : T_{t_{i-k},t_i} \rightarrow T_{t_{i+1},t_m}$ *that describes the behaviour of the user at time* $t_i$.

This definition combines all parts that constitute the problem of adaptively computing future contexts in dynamic ubiquitous computing environments [22].

## 2.4 Data Abstraction Level for Context Prediction

Since context has several data abstraction levels, we have to decide on which abstraction level the context prediction task is to be implemented. As raw sensor data is not yet in a normalised representation, several complications in context processing would be introduced if prediction was based on raw sensor data. We therefore suggest not to utilise raw-data for context prediction.

In the literature context prediction is usually based on high-level context information (see for instance [1, 9, 15, 23, 24, 25, 26]). These authors first interpret the low-level context to obtain high-level context information. Afterwards the prediction is based on the observed high-level contexts. This approach is appealing as long as the number of high-level contexts is low. Compared to the high number of combinations of low-level contexts from all available sensors the set of high-level contexts in typical examples is considerably small. The requirements for the applied prediction method are therefore low. This of course changes if the number of high-level contexts rises in more complex scenarios. Also, prediction based on high-level context information has vital restrictions due to a reduced knowledge about the context itself [27]. We therefore propose to base the prediction procedure on low-level context information (cf. figure 2).

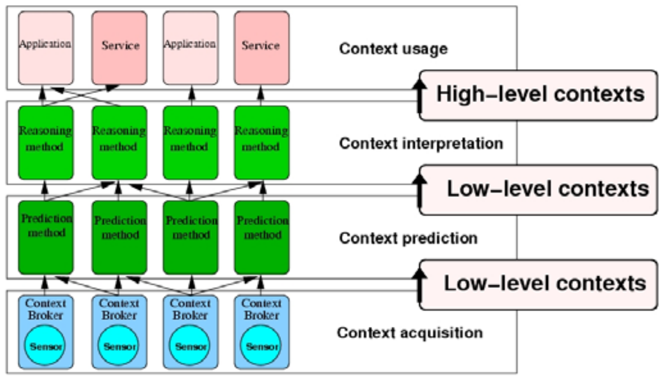


Figure 2: Context prediction based on low-level context elements.

We discuss issues on the context prediction accuracy that originate from the utilisation of data at various abstraction levels in the following sections.

## 3. AN ARCHITECTURE FOR CONTEXT PREDICTION

We introduce a general architecture for context prediction that was presented in [28]. This architecture is integrated into the Framework fOr ConteXT awaRe cOmpuTing (Foxtrot). Foxtrot follows a service oriented design with components that can be distributed between various mobile devices.

### 4.1 Foxtrot

We present an architecture for context awareness that is inspired by our main research interests and that serves as the basis for current and future research projects related to context awareness. We focus on a modular design and extensibility. This includes the possibility to easily exchange modules, if possible even at runtime.

The architecture is supposed to be executed in a distributed environment. Consequently the deployment of components of the architecture, discovery and network communication are required. To meet these requirements, we adopted a service oriented architecture approach and implement all functionalities as interacting, loosely coupled services. These services were built using the FAME$^2$ [29] framework. FAME$^2$ enables adding, removing and updating of services at runtime, which eases the maintenance of the system. Additionally it allows us to integrate several service discovery technologies and communication protocols to access arbitrary services from any other device reachable.

FAME$^2$ is a middleware that abstracts from the details of various computing platforms and provides a common interface regardless of the computing device to applications and services. FAME$^2$ is currently running on a multitude of computing platforms including mobile phones as the nokia communicator, smart phones as the MDA IV, pda's as the iPAQ and tablet PCs, notebooks and desktop computers running a windows or linux operating system.

The processing of context information in Foxtrot is a hierarchical process that is composed of three main parts. The first is the context acquisition part in which the sensor information is acquired, the second is the context processing part, in which the context information is further processed and the third step is the context usage part, in which the context information is utilised by applications and services. This process is illustrated in figure 5.

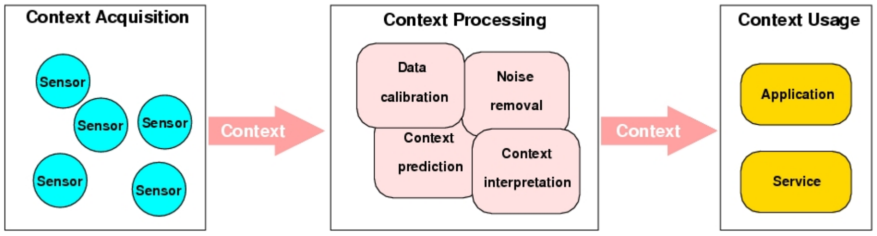


Figure 5: Architecture for context prediction.

The context processing part of this process may be furher divided into various processing steps.

### 4.1.1 Sensors

We consider sensors as atomic information sources for our architecture. Since several authors have varying definitions of sensors, we quickly recapitulate our definition here. Basically, a sensor is some piece of hardware or software that provides information on the environment. Humans or animals are not considered sensors but might trigger and influence sensor outputs. We distinguish between hardware sensors and software sensors. Hardware sensors are physical devices that react to stimuli from the physical environment and provide a software interface to publish notification describing these stimuli. Hardware sensors might for example measure the temperature, the light intensity or the humidity. Further hardware sensors are a computer keyboard or a mouse that monitor the user input or a fingerprint reader.

Software sensors are applications that react on software generated stimuli and that output a software generated notification describing this stimuli. Examples of software sensors are a calendar, an address book or an application the user is interacting with.

Similar to the Context Widget approach proposed by Dey[19], we encapsulate sensors by software components that provide a uniform interface between sensors and higher layer components (cf. figure 6).

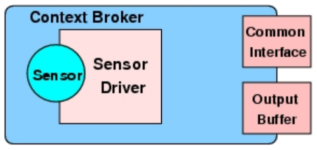


Figure 6: Conceptual design of a context broker.

We will refer to these encapsulating components as context brokers and assume that context brokers can be arbitrarily distributed on computing devices in the UbiComp environment. Pre-eminent components a broker consists of are a driver for the sensor encapsulated and a buffercontaining sensor output values. The context broker hides the sensor specific interface from higher architectural layers and provides a standardised interface to enable dynamic sensor discovery and removal in a flexible ubiquitous computing environment.

Since sensor measurements are stored in the buffer, the context broker can publish a sensor output at any time requested by simply providing the last value measured or by extrapolation.

### 4.1.2 Context processing

A context processing step processes context information. It is constituted from one or more processing modules, a common input interface and an output buffer that serves for communication with other modules in the architecture. Typical processing modules are noise removal, data calibration, context reasoning and context prediction. Processing steps can be distributed among devices in the environment. The communication and coordination of the data flow between various processing steps is established via the publish-subscribe paradigm. Figure 7 depicts an exemplary processing step.

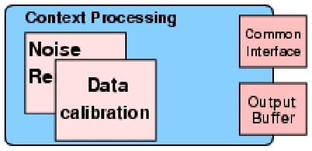


Figure 7: Conceptual design of a processing step.

### 4.1.3 Applications

Basically, the only requirement for applications that acquire context information from Foxtrot is the Foxtrot-interface. Applications subscribe to processing steps in order to receive the computed output of these processing steps in an event-based communication.
Furthermore, if an application shall be executed on arbitrary devices, a second requirement is the compatibility with FAME$^2$. To ease the development of services that comply with FAME$^2$ an eclipse-plugin has been developed at our chair. The adaptation of applications to FAME$^2$ is therefore straightforward.

## 4.2 Context Prediction Architecture

We insert the context prediction architecture between the context interpretation and the context acquisition layers (c.f. figure 2). Observe however, that due to the general definition of input and output data as context time series, the architecture could also be utilised for context prediction on high-level context elements (i.e. between the interpretation and the application layer). In section 5.2 we utilise this property in order to compare high-level and low-level context prediction schemes in simulation scenarios.

The architecture is constructed from four basic modules; a learning module, a context prediction module, a rule base and a context history. We group these modules into two processing steps. The learning method, the rule base and the context history are implemented together in one processing step, while the prediction module constitutes the main part of the second processing step. Furthermore, these modules interact with further processing steps in the context acquisition layer and the context interpretation layer (cf. figure 8).

All functional entities in the architecture are represented as processing steps which can be distributed on devices in the environment.

Context prediction based on low-level contexts is considered as one pre-processing step. Therefore, context interpretation, feature extraction and other pre-processing operations are not managed by the context prediction architecture. An illustration of an exemplary context data-flow can be found in figure 9.

Low-level contexts obtained from context brokers in the context acquisition layer may be further processed by context processing steps before they are utilised by applications.

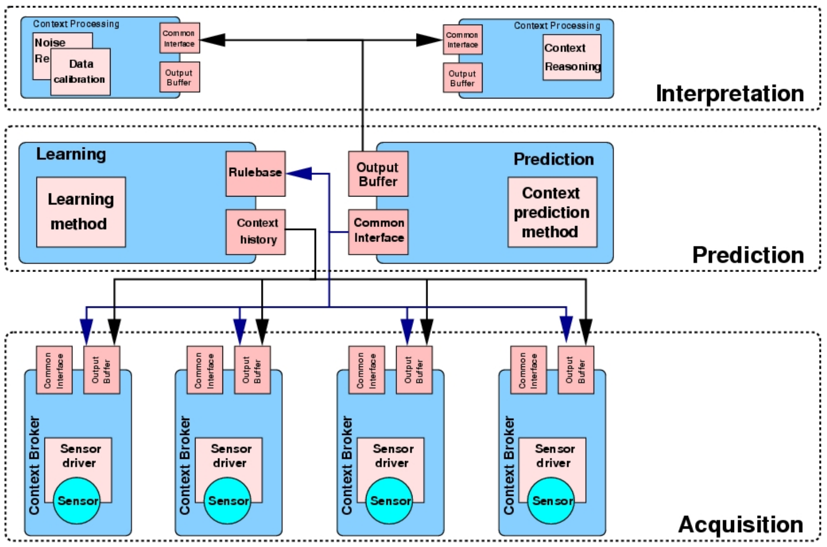


Figure 8: Our proposal of a general architecture for context prediction.

The context processing steps might process low-level contexts in a context prediction component or might interpret low-level contexts to high-level contexts or also process high-level contexts.

The figure also shows that not all data has to be utilised by the prediction component. For ease of presentation we represent the context prediction component in one context processing step in this figure.

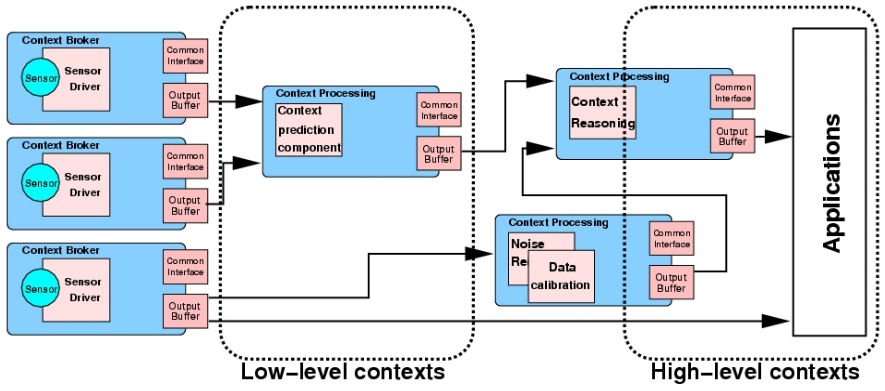


Figure 9: Placement of the low-level prediction component in the context data flow in a distributed environment.

### 4.2.1 Context history

We require the context prediction architecture to implement a continuous learning procedure that adapts to possibly changing user habits. We therefore require a context history that is constantly updated. The context history represents the ’memory’ of the prediction algorithm which is utilised to extract rules describing the typical behaviour of the user. The context history time series is fed by all context sources available in the context acquisition layer (cf. figure 8).

For each sensor attached to the architecture we track a sensor description and a time series of low-level context elements in the context history. The maximum length of the tracked time series is dependent on user preferences, implementation decisions or on memory constraints of the user device.

The context history is closely coupled with the learning method since it provides the input for this method. In order to protect the communication link between the learning method and the context history we integrate the context history in one processing step together with the learning method. The context history can be considered as the interface of the learning method to the observable environmental features.

### 4.2.2 Rule base

The rule base contains the rules that the learning algorithm constructs to guide the prediction module. The language in which these rules are represented is dependent on the prediction algorithm that is to access the rules for prediction tasks. Depending on the prediction algorithm utilised, various possible representations of varying abstraction level are feasible for these rules. Some examples are

- The time series of observed context elements
- Typical patterns in the time series
- Transition probabilities related to context changes or sequence changes
- A function describing the time series evolution

The rules in the rule base might have a weight attached in order to express the importance of the rule. We propose to keep the rule base on the user device since the rules generated are closely related to the context evolution the device experiences. Consequently we integrate the rule base together with the context history and the learning method in one processing step.

### 4.2.3 Learning component

The learning method has to support continuous learning in order to adapt to changing user habits or environments. The learning module therefore constantly monitors the time series stored in the context history and eventually uses some or all of these time series to construct and update rules that are then stored in the rule base. Since we do not propose a specific implementation the only requirements for the learning method are the interface specified by the context history and the description language for rules in the rulebase which is defined by the type of the prediction algorithm.

We describe the ability of the algorithm to adapt to a changing environment with the learning frequency. The learning frequency describes how frequently the method screens the context history in order to find new prediction rules. With a higher learning frequency hence the processing load of the learning module increases. However, an algorithm with a high learning frequency will more often update its rule base. A rule base that is more frequently updated will in general describe the current user habits more accurately.

### 4.2.4 Context prediction component

The actual context prediction is done by the prediction module. This module accesses the rule base and the momentary sensor data provided by the context sources in the context acquisition layer.

With the information from these two sources, the task of the context prediction method is to find the context sequence that most likely follows the most recently observed context time series. We had to take a design decision regarding the acquisition of recent context information in the context prediction component. Either the prediction component utilises the context history of the learning component or it provides a separate history of recent contexts on its own.

We propose to provide a separate history of context evolution in the processing step of the prediction method since the prediction algorithm might require another representation for the observed context time series and since it might utilise only a subset of all available context sources, as some context sources might be considered unimportant or unreliable.

By providing a second history of contexts in the prediction component, this prediction-algorithm specific complexity is hidden from the learning component. Furthermore, with this design various prediction components might utilise the same learning component.

## 4. SIMULATIONS

In order to compare context prediction algorithms we apply a Markov approach, an ARMA algorithm and an alignment prediction approach to various sets of sampled data.

## 5.1 Prediction of windpower samples

Wind is one of the inexhaustible energy sources humans have access to. In order to utilise this power, huge wind farms are built that consist of a multitude of windturbines. The amount of power available from these parks is constantly fluctuating and heavily dependent on the wind power. Since the power supply system is not capable of dealing with these heavily fluctuating power curves, methods that predict the wind power are required to schedule the power expulsion of other power sources the in order to balance the power level.

### 5.1.1 Simulation Scenario

The data set contains wind power samples from a wind farm in Germany and was recorded from February 2004 to April 2005 in an hourly fashion. We utilised 3/4 of these samples as training data and the remaining part for the simulation. The time series describing the wind power values contains real valued samples that range from 0 to 23.

We apply two context prediction algorithms to the data set. Namely a Markov approach of order 6 and the alignment prediction method. Both algorithms have their context history size (number of observed contexts stored) restricted to 6 hours. We complete several simulation runs in which we change the prediction horizon of the algorithms. Prediction results for prediction horizons ranging from 1 hour to 20 hours are obtained.

Furthermore, we utilise an ARMA approach with an unconstrained context history as upper bound. For comparison, we apply a modified version of the ARMA algorithm (ARMA-100). For ARMA-100 the context history size is restricted to 100 hours. To give an estimate on the accuracy of the prediction we calculate the RMSE and BIAS for every algorithm as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(p_i - d_i\right)}{n}} \qquad BIAS = \frac{\sum_{i=1}^{n} | \, p_i - d_i \, |}{n}.$$

In these formulas, $p_i$ depicts the predicted value at time *i* while $d_i$ is the value that actually occurs at time *i*.

### 5.1.2 Simulation Results

The results of the simulation regarding the RMSE values are depicted in Fig. 10.
As you can see from the figure, both ARMA and Markov methods are nearly identical for short prediction horizons while the alignment prediction approach performs worse. However, with prediction horizons that exceed the context history size (6 hours and more), the alignment prediction algorithm outperforms both the Markov and the ARMA-100 implementations. Still, it does not reach the same accuracy as the unrestricted ARMA approach.

When comparing the ARMA with the alignment approach, we have to consider the different resources both acquire. While the ARMA approach has access to the complete context history, the alignment approach utilises, apart from the training data, only a context history of six hours. As can be seen from the figure, with a context history restricted to 100 hours the ARMA-100 approach is already worse than the alignment approach for higher prediction horizons. If the context history is further reduced (e.g. to 50 hours and less), the simulation has shown that the ARMA approach is by far worse than the alignment algorithm for all prediction horizons.

Results for the BIAS metric are similar. The major difference to the RMSE results is that the Markov algorithm performs marginally worse and that the alignment prediction algorithm outperforms the Markov method already at a prediction horizon of 5 hours. Furthermore, for higher prediction horizons the accuracy of the Markov approach becomes nearly identical to the ARMA-100 algorithm.

Summarising, we have observed that the alignment approach performs well on the data set for high prediction horizons, whereas for low prediction horizons the Markov and ARMA-100 approaches may be preferred.

While these results have to be backed by further investigations, they show that the alignment prediction approach is well suited to extract plenty of information even from short context histories.

This property makes the alignment prediction approach especially well suited for ubiquitous computing environments. Since the user behaviour might be described by a series of short characteristic context patterns, it is essential that the prediction algorithm is capable of extracting a maximal amount of information from these patterns. We will investigate this property in further studies on sampled user-context time series.

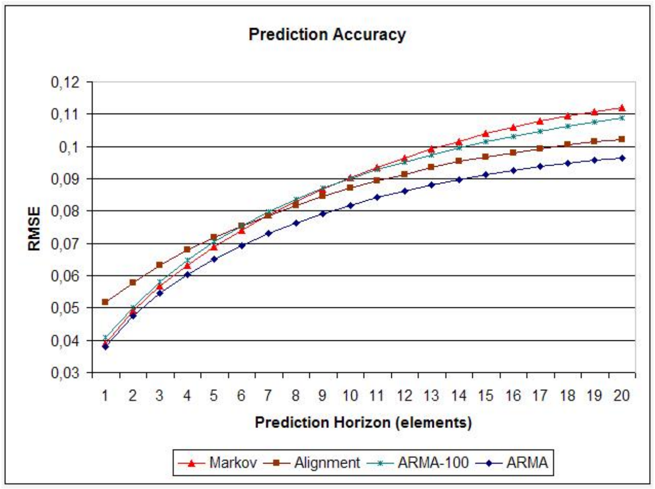


Figure 10: RMSE-values on the wind-power data set.

## 5.2 Location Prediction

We study influences of varying levels of context abstraction on the learning capabilities of high-level and low-level context prediction schemes on a sampled GPS trajectory of a mobile user. The sampling process lasted for about three weeks. The sampling hardware consists of a mobile phone and a GPS-receiver. The receiver is connected to the phone via bluetooth. A python script running day and night on the phone is used to obtain the GPS-information from the GPS-receiver.

This GPS data was utilised for our study. We took this two-step approach since we analyse high-level and low-level prediction schemes with varying input parameters. A real-time processing would have surcharged the memory and processing capabilities of the phone.

### 5.2.1 Simulation Scenario

The simulation consists of three weeks of GPS data from a mobile user. Every 2 minutes a GPS sample is taken. For times when no GPS is available (e.g. because the user is indoors), we assume that the last available sample best approximates the current position of the user. For the simulation we utilise the samples on a 20-minutes scale to reduce sequences of idle periods where no significant movement is observed. In an idle period it is best to extrapolate the current location of the user into the future. Therefore, if the observed sequence is dominated by idle periods instead of context changes, methods that extrapolate the current context into the future receive best prediction results. However, we do not consider this behaviour as learning and prediction, since the really significant context changes are missed although the prediction may be correct most of the time.

For the low-level context prediction we directly utilise the GPS-samples. For the high-level context prediction we define a total of 36 high-level locations as for example ’Home’, ’Bakery’, ’University’, or ’Market’. The high-level locations are specified by a GPS-centre-location and a radius. A default high-level location named ’Outdoors’ is applied when no high-level location matches a GPS sample.

As prediction algorithm we implement an alignment prediction approach [30] for low-level and high-level context prediction. The algorithm has a context history of 5 hours and a prediction horizon of 6 hours. All simulation specific parameters of the algorithm are identical for high-level and low-level prediction.

For evaluation of the accuracy we utilised the RMSE and BIAS metrics.

### 5.2.2 Simulation Results

We apply three simulations with varying amounts of learning time. In the first simulation no prior learning is allowed. In a second implementation, we utilise the first half of the data for learning purposes and in a third implementation, 3/4 e input data.

Table 2 depicts the RMSE and BIAS values for high-level and low-level context prediction schemes.

| | Low-level | | High-level | |
|---|---|---|---|---|
| | RMSE | BIAS | RMSE | BIAS |
| No learning | 0.7589 | 0.5759 | 0.9429 | 0.8891 |
| ½ learning | 0.5883 | 0.3461 | 0.9408 | 0.8852 |
| ¾ learning | 0.5879 | 0.3456 | 0.9402 | 0.8840 |

Table 2: RMSE and BIAS values for high-level and low-level context prediction schemes after various learning times.

For low-level context prediction a learning prior to the simulation improves the simulation result. However, the difference between the second and the third simulation is minor, since only few new patterns are observed in between. For high-level context prediction we observe that the overall result is worse than for the low-level case. While various reasons may account for this property [31], the learning gain is insignificant.

We observe this property also in figure 11.

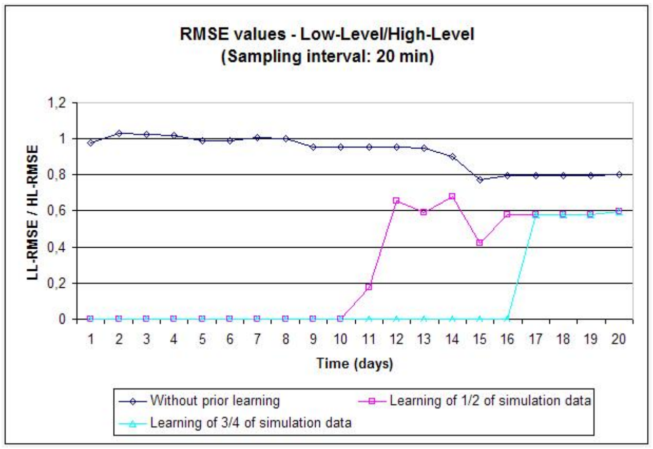


Figure 11: Comparison of RMSE values for low-level and high-level context prediction after various learning times

In figure 11 the fraction of the low-level RMSE-values divided by high-level RMSE-values is depicted. From the time-evolution we observe that the accuracy of the high-level context prediction scheme is above the low-level context prediction scheme for the first four days. This is due to the lower complexity of the high-level contexts. However, in the pace of the experiment, the low-level context prediction outperforms the high-level context prediction due to the higher information entropy of the low-level contexts. The depicted results contain two incursions at about the seventh and the fourteenth day. These are the weekends, where the GPS trajectory of the user significantly differs from his everyday trajectory. The low-level context prediction scheme is obviously better capable of coping with the new context patterns. Furthermore, the second incursion is stronger since the low-level context prediction scheme better utilises the information provided in the first weekend.

In the figure we further observe that the learning in advance of the simulation has a significant impact on the accuracy of the low-level context prediction scheme. For the BIAS-metric we have observed similar results.

Finally, we analyse the amount of memory required by the low-level and high-level context prediction schemes. For every typical context pattern observed, the alignment prediction algorithm stores a time series in a so-called rule base. As a measure we count the number of time series in the rule base and the number of context values in all time series (cf. figure 12).

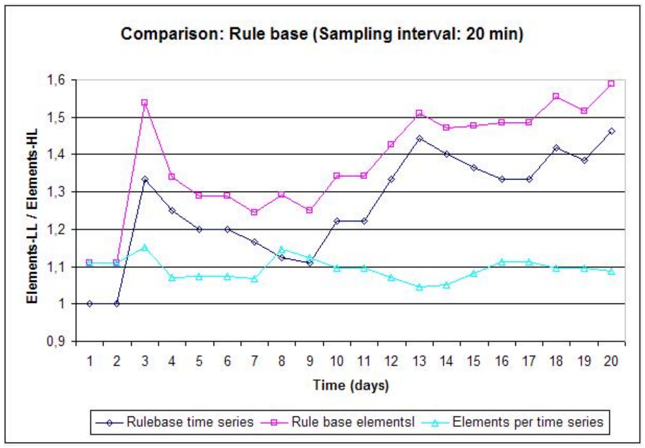


Figure 12: Contents of the rule base for low-level and high-level context prediction schemes for various prediction times.

Note that these figures also influence the processing load since the algorithm processes every single time series in the rule base in order to find alignments most similar to the observed time series.

We observe that the ratio of time series and context elements for low-level to high-level prediction schemes increases over time. After a simulation time of 20 days the low-level context prediction scheme stores roughly 1.5 times the amount of context values as the high-level prediction scheme. The number of context values per time series is more or less constant but approximately 1.1 times higher for low-level context prediction.

We therefore conclude that the low-level prediction scheme has higher memory and processing requirements although the simulation parameters are equal.

## 5. CONCLUSION

The ability to predict future contexts allows to improve the proactivity and quality of context-aware computing applications. In this paper we have introduced the basic principles of context prediction. Furthermore, the paper has described three different types of algorithms that can be used to predict future contexts. Foxtrot, an architecture for context-aware applications based on the FAME$^2$ middleware, has been introduced, and we have described how context prediction is implemented in this architecture. Finally, simulation results have compared different context prediction schemes and their accuracy. Additionally, we presented simulation results of the novel alignment prediction algorithm on a realistic context prediction problem. The investigations described in this paper show that a lower data abstraction level is beneficial to the context prediction task, since it contributes to a better adaptability of the prediction algorithm. However, the memory and processing requirements of low-level context prediction are higher compared to high-level context prediction schemes. Furthermore we have demonstrated that the alignment prediction algorithm is able to extract plenty of information from a short context history.

## ACKNOWLEDGEMENT

We would like to acknowledge partial funding by the German ”Bundesministerium für Bildung und Forschung” (BMBF) in the framework of the Wireless Internet and mik21 projects.

## REFERENCES

[1] R. M. Mayrhofer, “An architecture for context prediction,” Ph.D. dissertation, Johannes Kepeler University of Linz, Altenbergstrasse 69, 4040 Linz, Austria, Oktober 2004.

[2] M. Wooldridge and N. R. Jennings, “Intelligent agents: Theory and practice,” in *Knowledge Engineering Review*, vol. 10, no. 2, 1995.

[3] J.-P. Kreiss and G. Neuhaus, *Einührung in die Zeitreihenanalyse (in German),* Springer-Verlag, 2006.

[4] J. Brockwell and R. Davis, *Introduction to Time-Series and Forecasting,* Springer, 1996.

[5] G. E. P. Box and G. M. Jenkins, *Time Series Analysis forecasting and control,* Holden-Day, 1976.

[6] S.-H. Chun and S. H. Kim, “Impact of momentum bias on forecasting through knowledge discovery techniques in the foreign exchange market,” in *Expert Systems with Applications*, vol. 24, 2003, pp. 115–122.

[7] ------, “Data mining for financial prediction and trading: application to single and multiple markets,” in *Expert Systems with Applications*, vol. 26, 2004, pp. 131–139.

[8] E. Horvitz, paul Koch, C. M. Kadie, and A. Jacobs, “Coordinate: Probabilistic forecasting of presence and availability,” in *Proceedings of the Eighteenth Conference on Uncertainty and Artificial Intelligence*, July 2002, pp. 224–233.

[9] P. Nurmi, M. Martin, and J. A. Flanagan, “Enabling proactiveness through context prediction,” in *CAPS 2005, Workshop on Context Awareness for Proactive Systems*, June 2005.

[10] P. Brown, W. Burleson, M. Lamming, O.-W. Rahlff, G. Romano, J. Scholtz, and D. Snowdon, “Context-awareness: Some compelling applications,” in *Proceedings of the CH12000 Workshop on the What, Who, Where, When, Why and How of Context-Awareness*, 2000.

[11] M. C. Mozer, “The nueral network house: An environment that adapts to its inhabitants,” in *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, 1998, pp. 110–114.

[12] S. Sigg, S. Haseloff, and K. David, “Minimising the context prediction error,” in *65th IEEE semi-annual Vehicular Technology Conference (VTC2007-Spring) (to appear)*, April 2007.

[13] K. Leichtenstern, A. D. Luca, and E. Rukzio, “Analysis of built-in mobile phone sensors for supporting interactions with the real world,” in *Pervasive Mobile Interaction Devices (PERMID 2005) - Mobile Devices as Pervasive User Interfaces and Interaction Devices - Workshop at the Pervasive 2005*, 2005.

[14] M. Mulvenna, C. Nugent, X. Gu, M. Shapcott, J. Wallace, and S. Martin, “Using context prediction for self-management in ubiquitous computing environments,” in *Consumer Communications and Networking Conference (CCNC)*, January 2006, pp. 600–604.

[15] B. D. Davison and H. Hirsh, “Predicting sequences of user actions,” in *AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time–Series Analysis*, 1998.

[16] M. C. Mozer, “Neural net architectures for temporal sequence processing,” in *Time Series Prediction: Forecasting the Future and Understanding the Past (Santa Fe Institute Studies in the Sciences of Complexity XV)*, A. S. Weigend and N. A. Gershenfeld, Eds., 1994.

[17] C. Chatfild, *The Analysis of Time Series: An Introduction*, Chapman and Hall, 1996, vol. 5.

[18] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*, MIT Press, 1997.

[19] A. K. Dey, “Providing architectural support for building context–aware applications,” Ph.D. dissertation, Georgia Institute of Technology, November 2000.

[20] J. MÃ¤ntyjÃ¤rvi, “Sensor-based context recognition for mobile applications,” Ph.D. dissertation, VTT Rechnical Research Centre of Finland, 2003.

[21] B. N. Schilit and M. M. Theimer, “Disseminating active map information to mobile hosts,” in *IEEE Network*, vol. 5, no. 8, 1994, pp. 22–32.

[22] S. Sigg, S. L. Lau, S. Haseloff, and K. David, “Approaching a definition of context prediction,” in *Proceedings of the Third Workshop on Context Awareness for Proactive Systems (CAPS’07)*, June 18-19 2007, (to appear).

[23] K. Laasonen, M. Raento, and H. Toivonen, “Adaptive on-device location recognition,” ser. LNCS, no. 3001, 2004, pp. 287–304.

[24] D. Ashbrook and T. Starner, “Learning significant locations and predicting user movement with gps,” 2002.

[25] R. M. Mayrhofer, H. Radi, and A. Ferscha, “Recognizing and predicting context by learning from user behavior,” in *The International Conference On Advances in Mobile Multimedia (MoMM2003)*, vol. 171, September 2003, pp. 25–35.

[26] A. Ferscha, “Pervasive computing,” *Datenbank-Spektrum*, pp. 48–51, 2003.

[27] S. Sigg, S. Haseloff, and K. David, “The impact of the context interpretation error on the context prediction accuracy,” in *Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS 2006)*, July 17-21 2006.

[28] T. Loeffler, S. Sigg, S. Haseloff, and K. David, “The quick step to foxtrot,” in *Proceedings of the Second Workshop on Context Awareness for Proactive Systems (CAPS 2006)*, K. David, O. Droegehorn, and S. Haseloff, Eds. Kassel university press, June 12-13 2006.

[29] B. Wuest, O. Drögehorn, and K. David, “Framework for platforms in ubiquitous computing systems,” in *Proceedings of 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, 2005.

[30] S. Sigg, S. Haseloff, and K. David, “A novel approach to context prediction in ubicomp environments,” in *Proceedings of the 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2006)*, September 11-14 2006.

[31] ------, “Minimising the context prediction error,” in *Proceedings of IEEE 65th Vehicular Technology Conference VTC2007-Spring (VTC-2007 Spring)*, April 22-25 2007.