The Quick Step to Foxtrot

Tino Löffler, Stephan Sigg, Sandra Haseloff, and Klaus David

Chair of Communication Technology,
Department of Electrical Engineering / Computer Science,
University of Kassel,
D-34121 Kassel, Germany;
phone: +49 561-804-6446, fax: +49 804-6360
comtec@uni-kassel.de

Abstract. We propose Foxtrot, a highly flexible architecture for context aware systems that adapts to the versatile requirements of ubiquitous computing environments. To further enhance the gains of the architecture we introduce a context prediction paradigm that incorporates the possibility to increase the accuracy of proactive context aware applications. To utilise this potential we propose a novel context prediction algorithm. Additionally we introduce a methodology for designing context aware systems. Our development guidelines and the corresponding software equip the system designer with powerful analysis tools that support the creation of systems with greatly improved reliability.

1 Introduction

Context awareness is the ability of an application or service to sense the context in which it is currently executed and to react accordingly. Context awareness plays a dominant role in modern software systems and is about to increase its influence through novel user adaptive applications. Consider for example the following scenario.

When Milinda puts on her context–aware wrist watch in the morning, it starts scanning the environment for external sensors or devices. In the kitchen it senses the availability of the CD-player as well as the humidity, temperature and light sensors attached to the kitchen window and obtains data from them. When Melinda positively answers the prompt for turning on the CD-player, the watch initialises to play stimulating music since it is to be a bright day. After breakfast, on the stairs under way to her car Melinda meets her new neighbour Tom and stops for a chat with him. Due to the context time line the watch has observed so far it senses that Melinda is heading to work. Consequently it reminds her at 8:34 that she will be late if she kept on talking.

In this short example several interesting features of upcoming context aware applications neatly stick together. These are the sensing and utilising of remote devices and sensors, the context interpretation and the context prediction.

We will in this paper propose an architecture for context awareness that is highly flexible in supporting several concepts for context aware computing and may also be distributed among several mobile and stationary computing devices.

We further give a short insight into our approach to the research topics context prediction and context aware system development. Both are considered to significantly improve the scope of context aware applications beyond their current horizon. This work is organised as follows. In Sect. 2 we will give a brief overview of the related work, in Sect. 3 we present our architecture for context awareness while in Sect. 4 a novel methodology to design context aware systems and to distinguish the vital sensors relevant to a specific context is presented. Section 5 proposes our approach to context prediction and introduces the prediction module in detail. Section 6 summarises our results.

2 Related Work

Since context awareness has been recognised as a promising research field, various architectures for context aware systems have been proposed. The Context Toolkit [1] provides support for context aware systems. Applications can subscribe to predefined context sources, which can be plain sources, aggregators and interpreters. With our module based approach Foxtrot (Framework fOr ConteXT awaRe cOmpuTing) we adapt the general idea of this concept and further add the ability to distribute all modules to various mobile computing devices in a ubiquitous computing environment. Foxtrot further supports the addition, removal or update of modules at runtime. This is accomplished by adapting our architecture to the Framework for Applications in Mobile Environment (FAME^2)[2]. FAME^2 enables applications and services to be distributed between various devices and to be added, updated or removed at runtime. Another architecture for context aware applications is proposed by Henricksen [3] who concentrates on context modelling and proposes a layered architecture. Although similar to our approach, Henricksen does not reach the same flexibility in ubiquitous systems. The Solar platform [4] represents a context aware architecture which employs a graph-based abstraction for context aggregation and dissemination. Our approach is, while not graph-based, similar to that of the Solar platform but additionally considers a new context prediction layer on top of the context acquisition step and adds the aforementioned runtime flexibility.

This is also the main difference to architectures proposed by other authors. The authors of [5–8] have proposed an architecture for recognising context from multiple sensors in the TEA and Smart-Its projects. While [9] also considers a context prediction layer, the authors propose to include it at a later stage in the context processing procedure. In Section 5 we will argue why it is beneficial to apply the context prediction in an earlier stage.

3 Architecture for Context Awareness

We present Foxtrot, an architecture for context awareness that is inspired by our main research interests and that will serve as the basis for current and future research projects related to context awareness. Since we want to utilise the architecture also for upcoming research projects we focus on a modular design

and extensibility. Parts of the architecture are subject to constant changes. This means that the possibility to easily exchange modules, if possible even at runtime, is critical.

Furthermore the architecture is supposed to be executed in a distributed environment, so the deployment of parts of the architecture, discovery and network communication are also required. To meet these requirements, we adopted a service oriented architecture approach and implemented all functionalities as interacting, loosely coupled services. These services were built using the FAME^2 [2] 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.

3.1 Logical Segmentation

We divide our architecture into the four functional entities *context acquisition*, *context prediction / preparation*, *context processing* and *context usage* as illustrated in Fig. 1. Every functionality incorporates modules that are explained in the following.

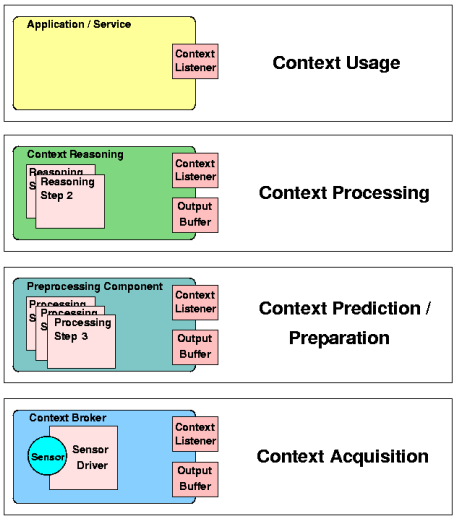

Fig. 1. Architecture for context prediction.

Context Acquisition The context acquisition part is responsible for acquiring context. It comprises context broker modules which read from attached sensors and convert the data into our context representation.

We refer to the output of any sensor as raw sensor data since it most probably needs further interpretation. Different manufacturers produce sensors with varying output even if the sensors are of the same class. This is because of possibly different encodings of the sensed information or due to a different representation or accuracy. After being processed by a context broker raw sensor data has become low-level context information. The low-level context information of two arbitrary sensors of the same class measured at the same time in the same place is identical with the exception of a possibly differing measurement accuracy, provided that both sensors are in good order.

Output values of a sensor are firstly transformed by the context broker from raw sensor data to low-level context information. The sequence of low-level context information attained from a sensor is stored to the output buffer of the context broker.

Context prediction / preparation Usually the low-level contexts processed by the context acquisition part of Foxtrot have to be further prepared for the following context processing step. Typical preprocessing modules in this step are the context prediction, the smoothening of values, the calculation of a mean value or other statistical methods. We refer to these operations as preprocessing steps. A preprocessing component may string various preprocessing steps and apply them to the low-level contexts in any order. The output of a preprocessing component is again low-level context information.

Context Processing The interpretation and aggregation of low-level context to high-level context information is done in the context processing part by a context reasoner. A designer of context aware applications may only with great effort make a funded decision based on low-level context information. He expects the low-level contexts to be aggregated to high-level context information that yields more meaningful information. A typical high-level context is for example *in a meeting* or *indoors*.

The context processing part of Foxtrot basically consists of one or more context reasoning components. Reasoning components may be stringed or may operate in parallel to other context reasoning components. It is therefore possible to directly compare the accuracy of different context reasoning components in real life applications or in simulations. We have currently implemented a reasoning module based on the RuleML language [10]. Future projects may reveal the need for a reasoning module that is not based on fixed rules.

Context Usage The high-level context obtained after the context processing step is transformed into an event and delivered to an event handler. Context aware applications can subscribe to the event handler and will be informed if a specific context occurs.

3.2 Support of Ubiquitous Computing

Since we believe that ubiquitous computing is one of the key application fields of any architecture for context awareness, we require Foxtrot to be highly distributable between various computational devices. Every module of Foxtrot can be distributed to various computing devices. It is therefore possible that the modules of a functional entity are spread among various computing devices. Furthermore every module may be available several times in the computing environment.

In a fluctuating ubiquitous computing environment a module may unexpectedly disappear because the computing device has moved out of reach. We expect the disappearance and reappearance of devices to be a common phenomenon in a flexible ubiquitous computing system.

To serve these demands we implemented our architecture as a FAME^2 compliant middleware [2]. This enables us to integrate several discovery technologies and communication protocols to access arbitrary modules from any other device reachable. Each module in Foxtrot is implemented to represent a service in FAME^2. Services in any FAME^2 compliant middleware may appear and dissappear or may be transferred from one device to another without the necessity to configure the device or the application itself.

Since all modules of one functional entity may be composed in virtually any order, the application designer might construct any desired application flow by choosing from a given set of modules. It is even possible that several different application flows are executed at the same time by the same modules.

4 Context-aware System Development

In Section 1 we already stated that context-aware systems are able to recognise or forecast context information. Based on this information they are supposed to adapt to the context automatically and to provide useful services and information to the user.

Context information can have different levels of abstraction. These levels range from measurable contexts like temperature to abstract, not measurable situations like meeting. Contexts, that are not measurable must be inferred from known context information. To stick with our example: the context meeting could probably be inferred from the time, the location, nearby people and sound patterns. These dependencies between measurable and not measurable contexts are often not obvious. For example, we have investigated the possibility to infer the location (outdoor/indoor) from temperature and humidity.

This leads to the conclusion that the selection of sensors and the knowledge about dependencies between contexts predefines the reachable reliability of our context detection.

Henricksen[3] proposes a situation abstraction supported by a context modelling language. The context to be detected is modelled from parts whose state is measurable. This approach works well if dependencies between sensor data and context are clear and sensor selection is predefined.

Clarkson et. al.[11] use an interesting method to determine Hidden Markov Models (HMMs) for their context detection. In their approach the user labels context states. The data is examined afterwards and the best performing HMMs are chosen for the detection system.

Our proposal of a methodology for designing context aware systems shares some basic ideas with the latter approach.

4.1 Methodology

Our aim is to design a methodology and a framework for the development of context-aware systems that helps programmers find dependencies between contexts and choose the right sensors for their system. We try to exploit experimental development procedures and data mining techniques to discover dependencies between contexts.

Previous to the development of the context-aware system we propose a three step experimental procedure to determine which sensors are appropriate and how the context can be inferred from their data.

Data acquisition In the first step we define the contexts that are to be detected and determine sensors that can be utilised depending on e.g. price, size and availability. These sensors are deployed depending on context and sensor type. The measured data is stored in a database. The occurrence of a predefined context is manually marked by the user.

Data interpretation In the second step we examine the dependencies between the sensor data and the manually marked contexts with data-mining techniques.

Inference logic design Based on the results of the data interpretation step the third step in the experimental procedure is to design the inference logic depending on the applied inference mechanism, i.e. defining rules or building a baysian network.

4.2 Framework

Beside the procedure and the guidance for the development of context-aware systems we plan to support developers with a software framework based on the introduced context-aware architecture. It will consist of tools supporting an easy integration of new sensors into the architecture, logging to miscellaneous data bases and analysing the data with various data-mining algorithms.

4.3 Benefits

We argue, that context inference and hence context-aware systems can be made more reliable if they are created following our approach.

The selection of sensors can be tailored to the context and the inference mechanism are backed by statistical data, whereby context detection errors are minimised. Deploying only sensors needed for context detection improves the system performance, because only needed sensor data is analysed.

5 Context Prediction Based on Low–Level Contexts

As already mentioned in section 3.1 we propose to include a context prediction module as one preprocessing component in the context preparation layer. Since context prediction in the literature is typically executed on high-level context information, we will in this section discuss the benefits and drawbacks of several context prediction schemes.

5.1 Context Prediction Schemes

In the literature context prediction is usually based on high-level context information (see for instance [12–16, 9]). 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.

However, the prediction based on high-level context information has vital restrictions due to a reduced knowledge about the context itself.

Reduction of Information. Some of the information contained in a low-level context is lost when transformed to a high-level context since the transformation function is typically not reversible. If the sampled low-level context information suffices to conclude a high-level context we can obtain this context at any time in the future provided that the low-level context information is still available. Once we abstract from low-level context information to high-level context information we cannot unambiguously obtain the low-level contexts we abstracted from. A time series of observed contexts consists of several samples from various sensors. The associated low-level contexts may indicate some general trend. However, the high-level contexts may mask this trend to some extend due to the higher level of abstraction.

Reduction of Operators. The only mathematical operator applicable to high-level contexts is typically the operator '='. All prediction methods that require other operators are not or only with additional computational overhead applicable to high-level context information. Popular context prediction methods therefore implicitly support non–numerical contexts but do not profit from contexts that provide additional information. The number of prediction methods suitable to low-level contexts is therefore larger than the number of prediction methods appropriate for prediction on high-level context information.

Reduction of Certainty. The prediction accuracy may be decreased when prediction is based on high-level context information. This is due to the different sequence in which the context prediction and context interpretation steps are applied. Let $P_{\text{acquisition}}$ be the probability that no error occurred in the context acquisition step. Furthermore $P_{\text{interpretation}}$ and $P_{\text{prediction}}$ are the probabilities that

no error occurs in the context interpretation and the context prediction step respectively. We write $P_{\text{prediction}}(i)$ if we want to address the probability that the context element at time t_{0+i} is predicted correctly. Assume that the prediction method bases its decision on a context history of t time series elements and predicts l future contexts. In the case of low-level context prediction each of the time series elements is composed of m low-level contexts. For high-level context prediction each time series element is represented by one high-level context. If the prediction is based on the low-level contexts, the probability that the predicted context element at time t_{0+i} is correct is

$$P_{\text{low-level}}(i) = P^{m \cdot t}_{\text{acquisition}} P^{m}_{\text{prediction}}(i) P_{\text{interpretation}} \ . \tag{1}$$

If the prediction is based on high-level contexts the probability, that the predicted context element is correct is

$$P_{\text{high-level}}(i) = P^{m \cdot t}_{\text{acquisition}} P^{t}_{\text{interpretation}} P_{\text{prediction}}(i) \ . \tag{2}$$

Comparing these results we obtain

$$\frac{P_{\text{low-level}}(i)}{P_{\text{high-level}}(i)} = P^{m-1}_{\text{prediction}}(i) \cdot P^{1-t}_{\text{interpretation}} \ . \tag{3}$$

Provided $P_{\text{interpretation}} = P_{\text{prediction}}$, prediction on low-level contexts leads to the higher accuracy compared to prediction based on high-level context information if the number of considered context elements from the context history is less than the number of low-level contexts in one time series element. Otherwise the ratio of $P_{\text{interpretation}}$ to $P_{\text{prediction}}$ further influences the prediction accuracy of low-level and high-level context prediction.

Reduction of the Long Term Accuracy. Provided that a sensor is in good order, the low-level context information reflects the actual situation with respect to the measurement inaccuracy. We therefore argue that low-level contexts are of higher credibility than high-level contexts. By interpreting low-level context information to high-level context information we take a guess based on the information that is available in the current situation. Since the context of the user is also dependent on information that is not measurable by sensors [17], we cannot exclude the possibility to misjudge the current context.

A prediction based on high-level context information might therefore be based on erroneous information. This does not only affect the instantaneous prediction accuracy but may also affect the long term prediction accuracy.

5.2 Time Series Prediction

For the above stated reasons, we base context prediction on low–level context information. In order to utilise the additional information on the context elements we use a time series prediction method that implements an alignment search

method. The observed context time series is aligned with every time series in a data base of typical context time series. By finding the optimal local alignment we also get the most probable predicted time series. For a decent study on alignment methods we refer to [18].

6 Conclusion

With Foxtrot we have proposed an architecture for context awareness that incorporates a high flexibility due to its modular structure. Because of its service oriented approach it is especially suited for highly flexible ubiquitous computing environments. Novel context aware concepts are easily adapted by Foxtrot even at runtime since the architecture is based on FAME^2. In our current research projets we study the impact of low-level prediction methods as well as context modelling techniques. We have argued that low-level context prediction has significant benefits to high-level prediction methods. Especially the prediction accuracy may be improved, while also novel prediction methods may be applied. We proposed a novel context prediction method that utilises the additional knowledge available for low-level contexts. To further guide the application designer, our proposed methodology for designing context aware applications provides powerful tools aiding the development process.

References

1. Dey, A.K.: Providing architectural support for building context-aware applications. PhD thesis (2000) Director-Gregory D. Abowd.
2. Wüst, B., Drögehorn, O., David, K.: Framework for platforms in ubiquitous computing systems. In: Proceedings of 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC). (2005)
3. Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. In: PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), Washington, DC, USA, IEEE Computer Society (2004) 77
4. Chen, G., Kotz, D.: Context aggregation and dissemination in ubiquitous computing systems. In: WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, Washington, DC, USA, IEEE Computer Society (2002) 105
5. van Laerhoven, K., Lowette, S.: Real–time analysis of data from many sensors with neural networks. In: Proceedings of the fourth international Symposium on Wearable Computers (ISWC). (2001)
6. Schmidt, A.: Ubiquitous Computing – Computing in Context. PhD thesis, Lancaster University (2002)
7. Schmidt, A., Beigl, M.: 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)
8. Schmidt, A., van Laerhoven, K.: How to build smart appliances. In: IEEE Personal Communications. (2001) 66–71

9. Mayrhofer, R.M., Radi, H., Ferscha, A.: Recognizing and predicting context by learning from user behavior. In: The International Conference On Advances in Mobile Multimedia (MoMM2003). Volume 171. (2003) 25–35
10. http://www.ruleml.org: (2005)
11. Clarkson, B., Pentland, A., Mase, K.: Recognizing user context via wearable sensors. In: ISWC '00: Proceedings of the 4th IEEE International Symposium on Wearable Computers, Washington, DC, USA, IEEE Computer Society (2000) 69
12. Laasonen, K., Raento, M., Toivonen, H.: Adaptive on–device location recognition. Number 3001 in LNCS (2004) 287–304
13. Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with gps. (2002)
14. Davison, B.D., Hirsh, H.: Predicting sequences of user actions. In: AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time–Series Analysis. (1998)
15. Mayrhofer, R.M.: An Architecture for Context Prediction. PhD thesis, Johannes Kepeler University of Linz, Altenbergstrasse 69, 4040 Linz, Austria (2004)
16. Nurmi, P., Martin, M., Flanagan, J.A.: Enabling proactiveness through context prediction. In: CAPS 2005, Workshop on Context Awareness for Proactive Systems. (2005)
17. Barkhuus, L.: How to define the communication situation: Context measures in present mobile telephony. In: Context, Stanford, CA, Springer (2003)
18. Boeckenhauer, H.J., Bongartz, D.: Algorithmische Grundlagen der Bioinformatik. Teubner (2003)