Towards a Better Understanding of Context Attributes

Tobias Zimmer
Telecooperation Office (TecO), University of Karlsruhe
Vincenz-Priessnitz-Strasse 1, 76131 Karlsruhe, Germany
Email: zimmer@teco.edu

Abstract

The use of context as an input is one of the major characteristics of Ubiquitous Computing systems. This paper looks into the structural and the systematic features of context in Ubiquitous Computing environments and their possible influence on context computing and the design of context-aware applications. This paper gives a definition regarding an understanding of context and how it is derived at a systematic level. Context interaction is investigated by a consumer – producer schema that models the exchange of context among artefacts. Validity, relevance, reliability and context history are identified and discussed as relevant systematic attributes of context.

1. Introduction

The use of context as an input is one of the major characteristics of Ubiquitous Computing systems. Context is used in theses systems to make the application as unobtrusive as possible for the user and to benefit from the added value that can be provided by its implicit input. The main focus of this paper is the structure and the inherent features of context data.

1.1. Representation and Sharing of Context

Most of today's application scenarios make use of context data in a customized way. That is, the information regarding the interpretation of context data is hard coded into the system logic. This is problematic as context-derivations of this sort are not reusable by other applications in general. An application reusing such a context would have to be provided with knowledge on the basis of which the underlying data was interpreted. Alternatively a more general encoding could be used that is understandable by all applications throughout an environment that are interested in some context information.

When the information on how contexts are structured is encapsulated within the application itself the context information is not easily accessible throughout the given environment. This fact conflicts with the idea of Ubiquitous Computing environments in general. Such interaction environments can only exist if (context) information is easily exchangeable between applications that do not necessarily have any prior knowledge of each other. Therefore, it is essential to agree on a common system for representation, communication and handling of context in an Ubiquitous Computing environment.

This paper discusses the systematic features of context in Ubiquitous Computing environments and their possible influence on context computing and the design of context-aware applications for Ubiquitous Computing settings. The focus here lies with how the identified features can be used in the representation and communication of contexts.

In the next section a definition of what will be referred to as context in this paper will be given, as well as the motivation for how context is derived form sensor values and other context information. It then briefly introduces a simple but powerful producer – consumer schema for contexts that can be used to model context interaction. The third section is dedicated to the properties and features of context.

2. Context: Yet Another Definition

Context models such as TEA [1] form TecO and the context toolkit [2] as introduced by Dey et al., focus on the handling of the context "content", and do not cope with the underlying structure of the context data itself. This is a very good approach for the application layers of context processing systems. These models describe mechanisms for the formation of context from raw data and its derivation from other contexts. However, this is independent from the need to provide a structured representation of contexts in order to make them processable by a wide range of computational artefacts. Therefore, a multi-stage model for processing, communicating and handling context data in Ubiquitous Computing environments, which is independent of the used methods of context generation and aggregation, is in-

troduced. This model takes into account important attributes of context itself for being communicated and processed and thus can provide easily usable context information to applications, the network and consequently to the user.

2.1. What is Context?

Within Ubiquitous Computing, the term context typically refers to "any information that can be used to characterize the situation of entities" [2]. As in this paper we look more into the processing oriented aspects of context we will refer to this more technical aspect of context as *context data* to avoid a confusion with the more general and natural definition of *context* in Ubiquitous Computing. So context data can be viewed as some kind of sensor value plus an additional piece of meta-information that can be used for interpretation of the given sensor value. The important step from sensor data to context data is made by *interpreting* the sensor readings.

Definition 1 Context data *is a piece of information that is composed of at least one piece of sensor data and one piece of meta-information used to interpret the sensor reading.*

Depending on the level of abstraction, this initial piece of meta-information can even be the identity of the sensor that is necessary to interpret a voltage level e.g. as temperature. In the model introduced here we chose a more natural approach: Sensor values in this model are already merged with their measurement unit. This follows the common understanding of sensor readings and prevents unnecessary complication of the system. The processing of sensor data is well understood from widespread measuring and control technology applications. However, the handling of meta-information and context data often is not. For this reason it is necessary to agree on what is meta-information and how it can be structured to be used to generate contexts.

Definition 2 Meta-information *is information on the environment that has been provided from outside the system itself. It can not be sensed in any way.*

Examples for meta-information are the application domain or information regarding the structure and purpose of a device that is provided by the programmer of an application; e.g. the identity of an artefact.

Context data is represented in a multi-stage model, where on the first (lowest) stage we have pure sensor data and raw meta-information. At this stage both sensor data and meta-information exist completely unrelated. In the second stage sensor values and pieces of meta-information are merged to a so called "first order context" or basic context.

Definition 3 First order context *is context data that is directly derived from raw sensor data and meta-information in one step.*

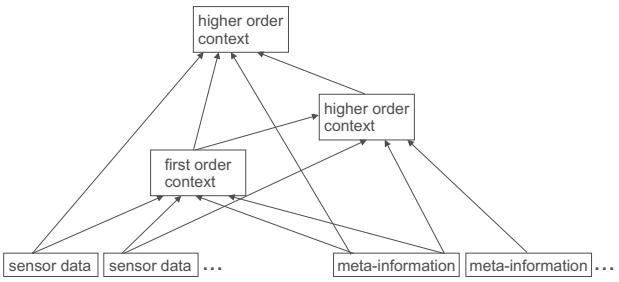

Figure 1. Multi-stage context model

First order contexts are characterized by a comparatively high reliability due to the fact that the generating device has gathered all the required information for the assembly of the context data by itself. It is therefore able to provide a good reliability measure for that piece of context data (a detailed discussion on the reliability of context data can be found in section 3.2). Within the higher stages of the model the "higher order contexts" that are compiled using sensor readings, some meta-information and at least one first order context, can be found. Higher order contexts as well can be derived form combining and interpreting some first or higher order contexts with additional sensor or meta-information in a recursive process.

Definition 4 Higher order context *is context data that is derived from at least one first order or higher order context and raw sensor data and/or some meta-information.*

Figure 1 shows a brief overview of the context data generation model.

In our model, every piece of context data has a type assigned. This type determines the identity of a piece of context data and thereby induces a context class of the given type.

Definition 5 *A* context class *is induced by the type of a given piece of context data. The context class has all features and attributes of a piece of context data with no values assigned.*

A context class is the prototype of a concrete instance of a piece of context data.

Definition 6 *A* context instance *is a piece of context data of a certain class with concrete values assigned to its features and attributes.*

In a context-aware environment two instances of a context class should always be distinguishable by the values of some features or attributes. As context data normally is not provided for exclusive use by one specific consumer there is no need for multiple identical instances of a piece of context data at any time.

2.2. Producers and Consumers

Artefacts in Ubiquitous Computing environments interact by communicating context data. An Artefact can either provide a piece of context data or use context data provided by other artefacts. When context data is provided it becomes an independent instance of the context class its type induces.

Definition 7 *An artefact providing a piece of context data is called* producer *of the piece of context data.*

When context data is used the instance is not necessarily destroyed. Multiple artefacts can use the same context data in parallel. With reference to the term ”context producer”, an artefact using context data is called ”consumer”.

Definition 8 *An artefact using a piece of context data is called* consumer *of the piece of context data.*

An artefact can be a context consumer as well as a producer at the same time, e.g. when deriving a higher order context form different input context data. Nevertheless, from the viewpoint of a specific context instance an artefact always acts as consumer *or* producer. This is to prevent single step recursions that lead to deadlocks in the context data processing.

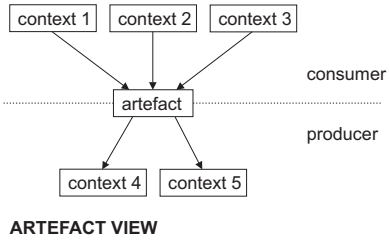

Figure 2. Producer – consumer schema

Figure 2 shows the context data interchange form an artefact centered point of view. The artefact acts as consumer and/or producer of context data. Context instances consumed are always distinguishable form the context instances produced by an artefact. This is the basis for introducing context history in section 3.3.

Figure 3 illustrates the information exchange among artefacts form a contexts viewpoint. Here a specific context instance is produced by an artefact and consumed by some others. The direct recursion in the production of context data is prevented by this. The shortest possible cycle consists of two artefacts that should continuously add to the context history, altering the instance of the context data. Still, this cannot completely prevent deadlocks in context processing.

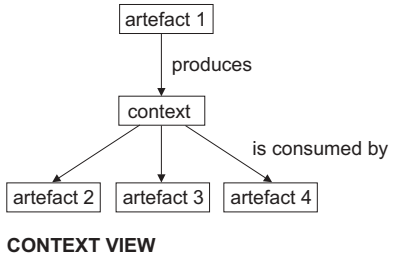

Figure 3. Producer – consumer schema

3. Attributes of Context Data

Context data in Ubiquitous Computing environments have some generic attributes that are independent from the context data content but should still be reflected in the representation of a given piece of context data. Knowledge of these attributes provides added value, not only to the application layer but, to all layers of a context-aware system. For example, the AwareCon [3] network running on the Smart-Its [4] [5] platform uses these attributes of context data to adjust its behavior to the needs implied by the context data communicated and the current situation.

Important generic context data attributes that are discussed in detail in this section are: *validity*, a coarse grain attribute and *relevance*, *reliability* and *context history* as more fine grained attributes. These attributes can be assigned to two classes: relative and absolute attributes. The value of absolute attributes can be provided by the producer of context data directly, whereas the value of a relative attribute has to be determined by the consumer on the basis of objective measures depending on the consumer’s application and overall situation. Both absolute and relative attributes are subject to the interpretation and assessment by the consumer.

3.1. Relevance

The relevance of a piece of context data depends on its age and the distance the context data was generated in [6] [7]. The time that elapsed since the generation can significantly reduce the relevance of a piece of context data for a consumer. This is due to the fact that sensor readings change over time and thereby continually invalidate context data which a certain reading is a part of. Thus those pieces of context data loose more and more of their relevance to applications over time.

The relative distance between the producer and the consumer of some context data also influences the relevance of a piece of context data. This can be illustrated by an example: At TecO we have a board marker equipped with a Smart-Its device. An application is running on that artefact that provides context information like ”the board marker is

in use". This context data (actually the change of context between "in use" and "not in use") is used to trigger a digital camera to take a picture of the whiteboard the marker was used on, when the writing stops. In this example the context data obviously is only relevant to applications near the whiteboard or respectively near the board marker since the application operating the camera is not interested in markers used on whiteboards in other rooms. Relevance is a relative attribute of context data. It has to be computed by the consumer on the basis of objective values like elapsed time and the locations of producer and consumer.

The age of context data is not to be mixed up with its history. Furthermore, the relevance of a piece of context data is independent of its reliability, for the reliability value of context data does not change over time and distance.

3.2. Reliability

Reliability of context data is one of the major issues when building context-aware applications, since the reliability of the applications directly depends on the reliability of the processed context data. For the use of context data in an Ubiquitous Computing system it is preferable that the context data is as reliable as possible, but still it is essential for the components processing context data to have a measure of how reliable a certain piece of context data is. One way to handle ambiguous context data is described by Dey et al. in [8], where a mediation process is introduced that makes use of user input to determine which context data is reliable. This approach is only feasible in systems where context data is mainly used on user-application level. When context data is used on lower layers of a Ubiquitous Computing system, explicit user input is no option due to two facts: Firstly, context dependent environments in most cases can not wait for user input. Secondly, user interfaces to all context-aware applications in the environment would be required.

This implies the introduction of a reliability metric for context data, that is able to provide the consumer of a piece of context data with additional knowledge on how to assess weight of the context data. The reliability of a piece of context data depends on the reliability of the input data that was used. The process of deriving new context data can produce only context data that is at most as reliable as the most unreliable piece of data that was used as an input. At this stage of research it is not yet clear to what extent statistical procedures like the ones used to decrease the measuring error in robust sensor fusion [9] can be applied to context data as well. Thus in most cases derived context data will not be as reliable as each of its pieces of source data.

Higher order contexts tend to be less reliable than first order contexts, because the producer uses first order contexts to compile the higher order context and has to relay on the reliability measure provided with the used input contexts to compute the reliability of the context data he produces.

For building reliable context-aware environments in Ubiquitous Computing it is necessary to develop a metric for measuring context data reliability in a way that allows assigning a reliability value to every piece of context data. Reliability is an absolute attribute of context data, as it is provided by the producer and can be used directly by the consumer as a basis of the decision on how to assess a piece of context data.

3.3. Context history

One of the outstanding characteristics of context is that in most cases it is dynamic [10]. When we look at common Ubiquitous Computing applications we find that some of them react on static contexts but most use changes of context to trigger events. The history of a context can be the changes in this particular context until now or it can be a sequence of contexts of the past that let to the actual context. The history of a context can be viewed as a representation of its dynamic character. In [10] Saul Greenberg points out that there are some factors connected to the highly dynamic character of context that complicate concluding from contexts and in the consequence making decisions and triggering events on basis of contexts. These are the dynamic itself, the potentially unknown intention of the user, the unreliability of contexts and the fact that different sequences of prior contexts can lead to the same context giving it a different meaning for the user or the consuming application. This last point mentioned is the history of the context. Thus context history is most relevant in designing context-aware applications for it can influence the meaning of a certain context. Context history is an absolute attribute of context, for it can only be provided by the producer of a context. Still its content is interpreted by each consumer, not necessarily leading to the same conclusions.

3.4. Validity

A piece of context data that is to be used to influence the behavior of a Ubiquitous Computing system obviously has to be valid. Validity of context data is a binary attribute. It determines whether a given piece of context data can be used in the processing of an application in a reasonable manner or not. The value of this attribute can depend on the relevance and reliability of the context data as well as its history as described above. The application consuming context data will have to decide on the validity of a piece of context data by its own, depending on its requirements. This makes the validity of context data a relative attribute depending on the consuming application.

3.5. Using Context Attributes in Applications

Context attributes apply to every piece of context data in a Ubiquitous Computing environment. They are inherent features of context data that add to the information carried as context itself. Most probably this list of relevant attributes is not complete, but it comprises the most generic ones that can add most value to a wide variety of context-aware applications.

One of the projects at TecO – AwareOffice [11] – focuses on augmenting office environments with Ubiquitous Computing technology. In this research project we develop a collection of applications that can easily benefit form the information provided by context attributes. E.g. the whiteboard application that was briefly introduced in section 3.1, assesses a relevance value to received board-marker-contexts on basis of the spatial distance of the context producer – the board marker – and the context consumer – the digital camera. The elapsed time since the emission of the context data is not that relevant to this application as it triggers on context changes form ”is writing” to ”is not writing” interpreting those as ”stopped writing”. The relevance measure in this case is more influenced by application specific factors: the camera only triggers when a certain amount of writing has occurred. Therefore, the relevance of the next triggering context data is increased when the ”is writing” context is received over a certain period of time. It is also increased every time a ”stopped writing” context is registered that was not relevant enough to trigger the camera. Thus the relevance threshold for taking a new picture of the whiteboard can be reached either when a certain amount is written in one go or when enough small changes have been made.

The board marker is augmented with a Smart-Its Particle Computer that derives the markers current state form 3D acceleration values. The algorithm used is subject to errors that are caused by misinterpreting certain movement patterns when someone holds the marker in hand and plays around or is gesticulating with it, like writing in the air while talking. These interpretation errors of the algorithm manifest as unreliability of the provided context. The algorithm therefore assigns a reliability value to every result representing the likelihood of an interpretation error.

Context history is used in this setting on the part of the camera application. As explained above the camera is triggered on context changes form ”is writing” to ”is not writing”. The ”writing”-context and ”not writing”-context are produced by the board marker and consumed by the camera. The ”stopped writing”-context is an internal piece of context data of the camera that it produces on basis of the history of the ”writing”-context. When detecting a change of context data – in most cases by the change of values of context instances of the same class – the use of context history is implicit. The consumer has to know about the history of a piece of context data to determine whether a change occurred between the reception of two instances.

4. Conclusion and Outlook

Utilizing the structure of context data can add to the value of context information in Ubiquitous Computing environments. As has been shown there are attributes common to all context data that can provide this added value when consequently exploited. To do so, it will be necessary to put more research effort into understanding the attributes introduced here and investigating additional ones. The work on context relevance of A. Schmidt in [7] can be a starting point. For assessing a reliability measure for context data a sound metric has to be developed and applied that takes into account the different modes of context data fusion and their influence on the reliability of the resulting context data. And a representation of context history has to be developed that accomplishes the same requirements as the representation of context data by means of interoperability.

References

[1] TEA. (1998) Technology for enabling Awareness (TEA). [Online]. Available: http://www.teco.edu/tea/

[2] A. K. Dey, G. D. Abowd, and D. Salber, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” *Human-Computer Interaction*, vol. 16 (2-4), pp. 97–166, 2001.

[3] M. Beigl, A. Krohn, T. Zimmer, C. Decker, and P. Robinson, “AwareCon: Situation Aware Context Communication,” in *Proceedings of Ubicomp 2003*, Seattle, USA, Oct. 2003.

[4] M. Beigl, T. Zimmer, A. Krohn, C. Decker, and P. Robinson, “Smart-Its – Communication and Sensing Technology for UbiComp Environments,” University of Karlsruhe, ISSN 1432-7864 2003/2, 2003.

[5] Smart-Its. (2003) The Smart-Its Project at TecO. [Online]. Available: http://smart-its.teco.edu/

[6] A. Schmidt, M. Beigl, and H.-W. Gellersen, “There is more to context than location,” *Computers and Graphics Journal*, vol. 33, no. 6, pp. 893–902, 1999.

[7] A. Schmidt, “Ubiquitous Computing – Computing in Context,” Ph.D. dissertation, Lancaster University, Nov. 2002.

[8] A. K. Dey, J. Mankoff, G. D. Abowd, and S. Carter, “Distributed Mediation of Ambiguous Context in Aware Environments,” in *Proceedings of the 15th Annual Symposium on User Interface Software and Technology (UIST 2002)*, Paris, France, Oct. 2002, pp. 121–130.

[9] R. McKendall and M. Mintz, *Data Fusion in Robotics and Machine Intelligence*. Boston: Academic Press, 1992, ch. Robust Sensor Fusion with Statistical Decision Theory.

[10] S. Greenberg, “Context as a Dynamic Construct,” *Human-Computer Interaction*, vol. 16 (2-4), pp. 257–268, 2001.

[11] AwareOffice. (2003) Awareoffice initiative at teco. [Online]. Available: http://www.teco.edu/awareoffice/