Modeling Contextual Information using Active Data Structures

Kevin Goslar and Alexander Schill

Dresden University of Technology
Chair for Computer Networks
{goslar,schill}@rn.inf.tu-dresden.de,

Abstract. In this paper, we present some ideas how to represent, handle, and integrate semantically expressive contextual information. Our approach enhances topic maps by more topic and association types and embedded code. Our data model is a comprehensive picture combining the knowledge of many context-aware applications. It is capable of performing automatic updates of the contained contextual information, computes higher-level information by itself, and routes context events to client applications. We present a context-aware front end of a car navigation system that uses our data model.

1 Introduction

The motivation of our work is to provide a semantically expressive information model for contextual data. Handling of contextual data is a crucial component for the success of Pervasive Computing (PervComp). The research community currently discusses some interesting approaches, but we still see unexploited potentials for the use of contextual information. Imagine the possibilities of all applications sharing their knowledge to create a large knowledge database about the users and their situation. Such scenarios require a contextual database (CDB), which stores and integrates contextual information from different applications. With such a CDB at hand, many problems and issues of PervComp would be implementable with justifiable effort. As *real world* we define the physical world, in which the humans live. A context-aware computer application, existing in the *computer world*, is aware of relevant aspects of the real world by monitoring *entities* (things) in the real world using sensors.

Sect. **??** provides the requirements for a contextual data model. Current approaches to model context information are reviewed in Sect. **??**. Sect. **??** provides the basic concepts of our approach and Sect. **??** gives the details. Sect. **??** concludes this paper.

2 Requirements for a Representation Format for Contextual Data

In this section, we describe the characteristics of pervasive environments and contextual data and show the resulting requirements for a specialized contextual data model.

2.1 Characteristics of Pervasive Computing

Pervasive computing scenarios have a number of specific characteristics, which need to be considered by a CDB. They are:

Heterogeneity: PervComp involves different devices, runtime environments, programming languages, paradigms, and applications. A CDB should be independent from those heterogeneous characteristics.

Individuality: Not all aspects of pervasive scenarios can be foreseen by the programmer of a pervasive application [**?**]. Individual requirements of individual users have to be maintained by the end user [**?**].

Offline use: Parts of the CDB need to be replicated to mobile devices for disconnected operation. Functioning replicas of CDBs need the data, semantic information about the data and logic to handle it.

Indirect Addressing: Entities are mostly identified by a circumscription like *set the temperature of* ***all rooms with no person inside*** *to 15 °C* instead of directly naming them with *set the temperature of* ***room 123*** *to 15 °C*, regardless of what is currently going on in that room, because this would be no context-aware behavior at all.

Many applications: PervComp scenarios involve many simultaneously running applications [**?**], which need to operate autonomously. Otherwise, the user would be disturbed with plenty of messages and questions.

Community: Many applications possess information about the context, which could also be useful to other applications and therefore should be integrated into the CDB [**?**]. This information should be integrated semantically into the database rather that ”stolen” from applications using software sensors.

2.2 Characteristics of Contextual data

Many researchers agree to the notion that the *context* of something is a collection of relevant parameters from the environment, which describe the situation of that entity [**?**,**?**,**?**]. This is a sufficient definition for our work. Contextual data has some unique characteristics, which need to be taken into account by a contextual data model.

Heterogeneity: Heterogeneous real world situations involve many types of entities, which must be modeled and handled efficiently by the CDB. Frequently used approaches, such as the relational data model [**?**], have limitations w. r. t. that aspect.

Complexity: Contextual data structures are highly interconnected graphs of objects, which can be too complex to be understood by human users and for the limited capabilities of mobile devices. It is necessary to break down the graph into smaller parts.

Distribution: Entities and sensors to monitor them are spread all over the real world. Contextual data emerges in different computer systems, making it necessary to integrate different data sources semantically into one ontology.

Data quality: Contextual information always represents imperfect assumptions about the real world [**?**]. We estimate the validity of contextual data based on data-specific quality indicators as it is proposed by [**?**,**?**].

Dynamics: To avoid permanent update operations, a CDB should store the information how to read the current value of fast changing contextual data from the environment rather than storing the current values themselves.

Mutability: The real world evolves and changes permanently [**?**,**?**]. Data types change and new data types occur. The data model should be extensible at runtime without the need to change, recompile or restart the database engine.

Unavailability: Higher-level knowledge is based on very detailed low-level information, which is not always immediately visible to the sensors. It is necessary to observe the user for a while until all necessary data is available in the CDB.

Privacy: Pervasive technology will capture and store confidential and private information. Therefore, we need security mechanisms as well as a data model, which is simple enough to be inspected by the human end users.

2.3 Requirements for a Representation Format for Contextual Information

In addition to the requirements that were derived from the characteristics of pervasive environments and contextual data we identified the following requirements for contextual data.

Formality: Only a well-defined data model allows non-ambiguous representations, which can be translated into the ontology of other applications or used by reasoning mechanisms [**?**].

Globality: Although a consistent model of all existing data is hard to implement and maintain [**?**], we need at least an unambiguous identification schema for contextual data. We model data in small modules, but address it globally.

Modularity, Reusability: Application specific data should be maintained and provided by the respective applications. Generic data structures should be reused to avoid structural redundancy.

Extensibility: Adaptation of the contextual data model to the changing environment is done permanently. The contextual database must be flexible enough to be extensible at any point with many new concepts.

History: Reasoning mechanisms need previous values of contextual data to recognize trends and extrapolate prospective values. The CDB should be able to reproduce snapshots of previous moments.

3 Related Work

The early context-aware research provided frameworks for pervasive infrastructure, i.e. sensors and communication issues. More recent research is becoming aware of the problem how to represent contextual information.

3.1 Object-oriented Approaches

Many earlier context-aware approaches like the *Context Toolkit* [**?**] model contextual information as classes in an object oriented programming language. These application-oriented frameworks encapsulate context-aware functionality like dealing with sensors. They mostly use static models of context with simple data types, which cannot be changed at runtime, and do not allow to compose a context model out of modules that are provided by different applications.

3.2 Topic maps

Power [**?**] suggests the use of topic maps to integrate contextual data located in different contextual databases. Topic maps allow defining relations between physical or logical objects located inside or outside the computer world. Power keeps the data stored at their original location and uses a metadata structure (the topic map) to interconnect them. He also mentions the idea to store context data inside the topic map itself, but he doesn't discuss this idea in detail. While realizing the need for more dynamic topic maps in order to handle the characteristics of contextual data, no details are provided how such an extension could look like.

3.3 Henricksen et al.

Henricksen et al. [**?**] propose a data format for contextual information based on semantic networks. Several types of associations are defined, e.g. distinguishing associations to data obtained from sensors or data provided by the user. This is relevant metadata for contextual information, but we need more precise information about the source of contextual data, e.g. if it was obtained from hardware or software sensors. Higher-level information is inferred by derived associations, which contain a derivation function that performs algorithms to compute new knowledge out of existing data. This is a powerful concept, but it is not determined where the algorithm of the derivation function is defined and when and how it is invoked. We see the data derivation function as a part of the data model and therefore store its code or its invocation inside the data model. Association types with quantitative qualifiers are also defined. Collection associations can occur more than once for each entity, a set of alternative associations represents alternative possibilities in the graph, and from a set of temporal associations only one association is valid in a given time interval. This allows modeling the past and the present together in one representation, but leads to very complex data structures, which show plenty of irrelevant information. The data quality is modeled using data-specific quality attributes.

This model provides useful concepts for a contextual data representation format. It provides means to visually represent logic in the data model, but only for very basic operations. This leads to complex circumscriptions of the intended structures. A modular concept is missing to break down the data structure in simple, reusable pieces as well as a concept to semantically integrate data from different contextual applications and databases. The model integrates several dimensions of contextual data (i.e. history, adaptability, dependency), in one representation, which will lead to complicated representations in large application scenarios.

4 Basic Concepts of our Approach

In this section we present ideas how to represent contextual data, which is used by more than one application. Individual context-aware applications in well-defined scenarios are implemented more effectively using existing approaches like [**?**].

Semantic net: A context model should reflect the real world without unnecessary abstractions. Opposed to AI approaches, which mostly depict the world as a collection of strange looking formulas (see [**?**]), a semantic network arranges the objects intuitively. A net-like data structure allows ”browsing” to the intended data (indirect addressing, see Sect. **??**).

Scaling: Human users need simple views of complicated data. We use simplified views of complex parts of the map to provide low-detailed versions of unneeded parts of the data structure.

Reusability: Contextual information can be used by more than one application. Our CDB has a generic, application independent core model, which can be extended with application-specific details (see Fig. **??**). The gray topics represent users and are the link to merge the global and the application-specific user information. The whole database is available to all applications. To reduce redundancy in the contextual data structure, applications can reuse models of other applications.

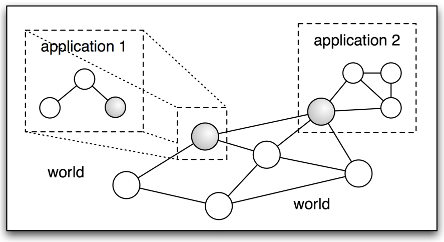

Fig. 1. global and application specific contextual information

Data together with code: To be truly independent from the current platform, database engine, and client application, a contextual data model must contain its own routines to compute all necessary data by itself.

Remembering the accruement of data: Since contextual data is imperfect [**?**] and has temporal as well as spatial localities [**?**], we need to keep the sources of contextual data in mind to assess the validity of the data. For example the CDB could work out contradictorily statements like *the user is currently in a meeting, and I know this because I have examined the schedules in her PDA* and *the user is currently driving a car, and I know this from the car computer system having identified the user* by deciding that the latter information is used, because it is based on more actual and accurate sources.

5 Our Solution in Detail

In this section we first describe our sample application called SCANav and then its underlying contextual database technology.

5.1 SCANav Application

We use our concept in a context-aware user interface for car navigation systems called SCANav, which guesses the intended destination of the driver by searching the emails, to-do items and schedules of the current user for appointment data, i.e. names of places together with statements expressing date and time. After checking if these appointments are useful as destinations, they are proposed to the driver when she enters the car. Users can also enter the name of a person or place into the system and the respective address is looked up in the user's or public address books.

It soon became evident that a user won't be able or willing to manually configure all context-aware devices like her car navigation system or her intelligent refrigerator. It is necessary to provide the user's data and preferences to all context-aware devices automatically. Furthermore, real world situations are too heterogeneous for hard coded applications. A user might use different types of electronic messaging, e.g. email, instant messages, or electronic notes, which need to be known to our application as ”electronic mail”. A dynamic, intelligent, adaptable representation of the world is necessary.

5.2 Context Maps

We decided to base our contextual representation on topic maps [**?**]. Topic maps are a relatively simple, easily understandable semantic net. Their original purpose is to provide information about any kind of real world or electronic entities, which is exactly what context-awareness is about. The representations of the real world objects in the topic map are called topics. These topics are interconnected by associations, which represent the relations between the real world entities. It is possible to store links from topics to their respective real world objects (these links are called occurrences) and to provide unique IDs for topics (called published subject identifiers), which can be used to address topics from other maps. The concept of base names allows providing several synonyms for a topic. Topic maps support locality by a concept called scope. A scope defines a semantic area in the topic map. Topics associated with a scope are visible only inside its respective area. This allows to distinguish homonymic topics in different contexts. In the following sections, we present extensions of topic maps (called *Context Maps*) to make them more useful for contextual data.

5.3 Universe

Topic maps as a metadata technology allow providing metadata, e.g. class definitions, for themselves using a ”class-instance” association class, but this is not mandatory nor well supported by the topic map standard. To provide a separate, reusable, formal model for contextual data, we invented a metadata structure called *universe*, which declares all topic and association types and their semantic meaning for a certain application scenario. A universe provides the background information that is necessary to understand the situations and processes in the respective application scenario. It includes ontological information as well as background information about the object and relation types that might occur there. This makes sure that different client applications, which use the context map as a source of information, interpret the information from this map in the same way. Every Context Map needs a universe as a foundation.

5.4 Dynamic Topics

Dynamic topics model very volatile entities, which occur in varying quantities and are stored in an external database. A dynamic topic is a placeholder that provides all necessary information to obtain the current value of the dynamic entities, which are instantiated for each request. Figure **??** shows a part of the SCANav context map, which

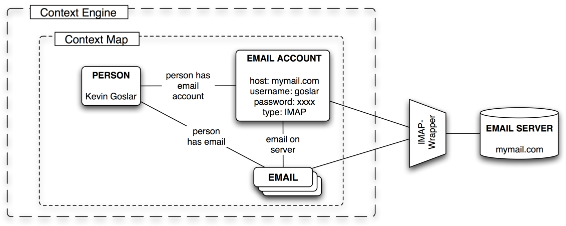

Fig. 2. a dynamic topic

contains a normal topic for the user, her email-account, and the dynamic topic ”email”. It furthermore shows an *access wrapper* to perform the access to the external email server. Whenever an association to the dynamic topic ”email” is used, a query to the underlying email account is done and the temporary topics representing the emails are created as shown in Fig. **??**. These topics are used as regular topics. After their lifetime has expired, the temporary topcs are deleted from the topic map. Only the dynamic topic is left in the topic map, resulting in the original situation in Fig. **??**. The lifetime of topics is usually represented as an attribute containing the lifetime of the entity in seconds. An approach that changes the validity of contextual data in dependency from their lifetime using fuzzy logic has been invented by [**?**]. This approach, however, is not used here for simplicity. Dynamic topics do not contain user-defined logic. They are simple instruments to model dynamic data. They are automatically configured with data taken from the topic representing the remote database, in our example the ”email account” topic, which is also connected to the access wrapper (see Fig. **??**). For more sophisticated functionality, we use intelligent associations (see Sect. **??**).

5.5 Complex Associations

Modularity is implemented by two different concepts. (1) Every application defines its data in an application specific map, which is merged into the global representation.

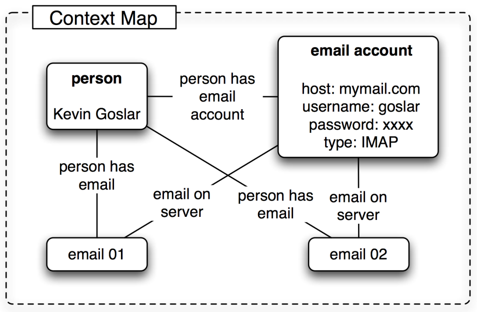

Fig. 3. temporary topics replaced the dynamic topic

(2) We model parts of the maps separately using *Complex Associations*. This association type provides two alternative representations for a relation between two topics: a simple, high-level view and a detailed, low-level view. The client of the Context Map engine can choose the representation in order to start browsing the map with a simple, high-level perspective and zoom into more detailed views as needed. An example is shown in Fig. **??**. It shows a normal association between a person and her car and

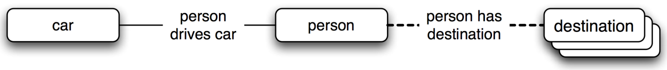

Fig. 4. high-level view of the SCANav context map, with a complex association

a complex association ”person has destination”, which is represented as a dotted line. This association is a high-level representation of the map shown in Fig. **??**. Apparently,

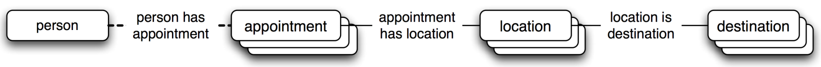

Fig. 5. replacement map for the complex association ”person has destination”

destinations of persons are determined by getting the persons appointments (with the association ”person has appointment”), scanning them for location descriptions (with the association ”appointment has location”) and deciding if that location is an appropriate target for a car (with the association ”location is destination”). Complex associations can be used recursively, e.g. the association ”person has appointment” in Fig. **??**, which is also a complex association, could also be replaced by a more detailed map, which describes that we compute appointments out of the user’s schedule and by text mining her email. Finally, we get the representation shown in Fig. **??**.

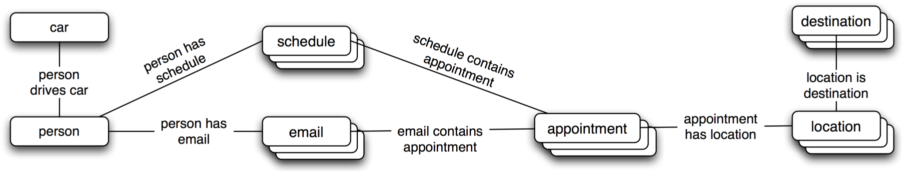

Fig. 6. the complete context map of SCANav

5.6 Intelligent Associations

In Fig. **??**, we see the steps to compute possible destinations of the current driver of a car. The single steps, for example in the association ”appointment has location”, which computes locations out of appointments, are performed using *intelligent associations*. An intelligent association contains a code block, which is executed every time the association is used. This code typically creates or changes the topic at the other end of the association or determines if the association is valid in the current situation. This enables the data model to change its structure at runtime, depending on the current situation. Intelligent associations can furthermore be used to do transformation operations on the data in the map, e.g. refine raw sensor data into higher-level information. Intelligent associations should not be used to provide the content of volatile topics, because this very common problem is addressed in a more specialized way with dynamic topics.

5.7 Automatic Interpretation and Routing of Events

Our contextual model does not just represent contextual data, but also detailed information about the relations between the contextual objects. This enables the contextual database to react to real world events automatically. Whenever a sensor reports a new value, its respective topic is updated. The database automatically recalculates the value of all other topics connected to this topic by following the topic’s associations. Since applications can subscribe for changes of a certain topic, all related parts of the context map are updated automatically and interested applications are notified whenever the environment changes.

6 Conclusion

We presented an approach to model and handle contextual data using an active semantic network called Context Map. It is based on topic maps and uses several new types of topics and associations to handle the characteristics of contextual data. In particular, we presented the following technologies:

Dynamic topics model volatile, fast changing data elements that occur in varying quantities. They are instantiated and filled with the current data from the context when accessed, Complex associations provide a modular concept to reuse components and simplify the modeling process as well as the navigation in the map. They allow simple, high-level views of the map, into which more topics and details for selected associations are faded.

Intelligent associations have an intelligent, adaptive behaviour due to their embedded code block. They can calculate the validity of the association at runtime or change their adjacent topics, thereby allowing modifying the data structure at runtime. They can also be used to implement transformation operations on data inside the map.

Our approach is a generic, self-adapting, application and implementation independent model of the real world, which can be extended with application-dependent contextual data. The result is a comprehensive representation of the real world, which combines and shares the knowledge of different context-aware applications. The framework is currently under construction. Our next steps are the evaluation of the framework w.r.t. its usability, expressiveness, performance, cost, scalability, and portability.

References

1. E. Codd. A relational model for large shared databanks. *Communications of the ACM*, 13(6):377–387, 1970.
2. A. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology, 2000.
3. A. Dey and G. Abowd. The context toolkit: Aiding the development of context-aware applications. In *Software Engineering for Wearable and Pervasive Computing*, 2000.
4. A. Fitzpatrick, G. Biegel, S. Clarke, and V. Cahill. Towards a sentient object model. In *Workshop on Engineering Context-Aware Object-Oriented Systems and Environments*, Seattle, USA, 2002.
5. K. Goslar, S. Buchholz, A. Schill, and H. Vogler. A multidimensional approach to context-awareness. In *7th World Multiconference on Systemics, Cybernetics and Informatics*, pages 229–234, Orlando, USA, 2003.
6. K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In F. Mattern and M. Naghshineh, editors, *Pervasive 2002*, pages 167–180. Springer Verlag, Berlin, 2002.
7. M. Klann. Eud-net's roadmap to end-user development. In *Workshop on End User Development*, pages 23–26, Fort Lauderdale, USA, 2003.
8. P. Nixon. Engineering context-aware enterprise systems. In *Workshop on Engineering Context-Aware Object-Oriented Systems and Environments*, Seattle, USA, 2002.
9. Topicmap.com website, 2004.
10. R. Power. Topic maps for context management. In *Workshop on adaptive systems for ubiquitous computing*, 2003.
11. A. Ranganathan and R. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, 2003.
12. A. W. Scheer. *Enterprise-Wide Data Modeling: Information Systems in Industry*. Springer-Verlag, Berlin, 1990.
13. A. Schmidt. *Ubiquitous Computing - Computing in Context*. PhD thesis, Lancaster University, 2002.
14. R. Wang, M. Reddy, and H. Kon. Towards quality data: an attribute-based approach. *Decision Support Systems*, 13:349–372, 1995.