Ontology-Based User Context Management: The Challenges of Imperfection and Time-Dependence

Andreas Schmidt

FZI Research Center for Information Technologies
Information Process Engineering
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
Andreas.Schmidt@fzi.de

Abstract. Robust and scalable user context management is the key enabler for the emerging context- and situation-aware applications, and ontology-based approaches have shown their usefulness for capturing especially context information on a high level of abstraction. But so far the problem has not been approached as a data management problem, which is key to scalability and robustness. The specific challenges lie in the imperfection of high-level context information, its time-dependence and the variability in the dynamics between its different elements. The approach presented in this paper presents a layered data model which structures the problems and is geared towards flexible and efficient query processing in combination of relational database and logic-based techniques. The techniques have been successfully applied for context-aware corporate learning support.

1 Introduction

Situation (or context awareness) has become a major topic in a wide range of research areas – among them mobile information systems, ambient intelligence, adaptive e-learning and knowledge management systems. Especially in information systems research, this expresses the insight that after the quest for making available vast amounts of information and for doing that efficiently, it is now the user who is the bottleneck. In order to find relevant information, the user needs to specify more precisely what she actually needs. But in many cases, the user is either not capable of doing that, or it drastically reduces the usability of the system – or both. This usage efficiency dilemma between selectivity on the one side and ease of use on the other side can be overcome by the system's awareness of the situation of the user. The system can then add transparently implicit assumptions of the user to her explicit queries or actions.

This idea sounds compelling, but closer inspection reveals that it faces fundamental challenges. Most of them can be traced back to the problem that the system cannot sense the usage situation (as the subset of the state of the real world relevant to the interaction with the system) directly, but has to rely on the usage context as a model for that situation (see fig. 1). This model is the result of a mapping which is highly imperfect in its nature (see also [1]):

– The mapping is **incomplete** as the system will never be able to capture all of the different aspects of the situation.

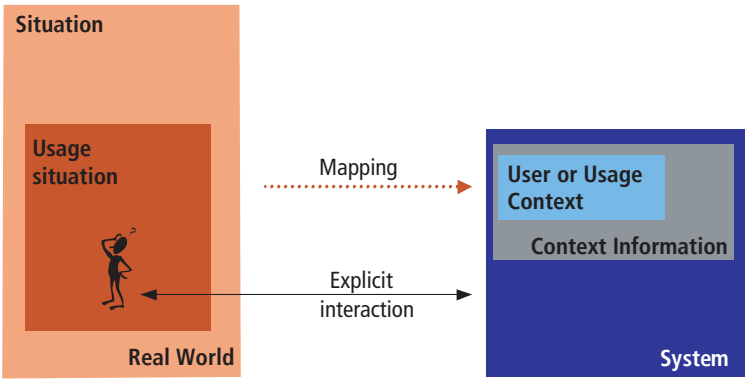

Fig. 1. Situation and Context: The origin of the problems

- The mapping is **uncertain** as the system has to rely on indirect methods and heuristics for eliciting context information from observable data.
- The mapping is **imprecise** as these methods yield only results with limited precision
- The mapping is **inconsistent** as a consequence of contradictions, resulting from different methods and their uncertainty and imprecision.

Especially for high-level context information [2], i.e. context information on a high level of abstraction as opposed to sensor-level information, this is aggravated by the problem of dynamics [3]. On the one side, it is often not possible to determine context on demand, but rather the system has to collect its pieces in advance *(asynchronicity of acquisition and usage)*. But on the other side, some parts of the context change often quite quickly, others are rather stable *(variability in the rate of change)*. Furthermore, efficient user context management requires a fairly deep *understanding of the context semantics*, especially when augmenting the collected context information, which is inevitable for high-quality context. This makes the provision of context information to applications a complex task that should be realized by a specialized user context management service – in analogy to other data management systems.

The requirement for understanding the context semantics calls for ontology-based approaches, which have proven their usefulness for incorporating a shared semantics in applications, but ontology management systems currently in use are not geared towards the pecularities of user context data, which is characterized by various forms of imperfection and a strong time-dependency both in terms of validity and reliability [4] combined with a high update frequency. In this paper, an approach is presented that is capable of representing and efficiently dealing with these challenges. The organization of the paper is as follows: in section 2, the requirements for user context data management will be analyzed and presented. In section 3 the layered data model is presented that is used to represent user context data. Section 4 will cover the issue of integrating ontologies into the data model. In section 5, some implementation issues will be dis-

cussed in the frame of a case study. The paper will close with a review of related work in section 6 and conclusions and outlook in section 7.

2 Usage Scenario

2.1 Scenario

The main guiding scenario for the user context management approach presented here was context-steered workplace learning [5]. In order to achieve the integration of work and learning processes, the learning support system does not rely on the user searching actively for appropriate learning material, but observes what the user is currently doing. Based on these observations and background knowledge on the competency requirements of elements of the user's situation like task, process, or role, the system then suggests appropriate learning programs which are compiled on demand from fine-grained learning objects. Additionally, the system can also suggest co-workers who are experts in a certain area, or who were in a similar situation recently. This approach of awareness of the work and learning situation can help to reduce the cognitive load of self-steered learning drastically. In this scenario, situation awareness has many different aspects which have been systematically analyzed in [6]) and can be divided along the different phases of the e-learning process: authoring, delivery (what, when, how), and execution.

2.2 Use cases

In order to support the scenario sketched above on a technical level, a service-oriented architecture of learning support services has been conceived within the project *Learning in Process*. The need of these services has formed the basis of the following use cases from which the requirements for the user context management approach have been derived:

- **Retrieve feature values for the current context**. The standard use case for context-aware application is the retrieval of certain context feature values which are considered relevant for adapting the system behaviour. A learning system can adapt the presentation to the technical context (broadband access, loudspeakers available) or the selection based on the current project context or mid-term interests and goals.
- **Check for certain feature values**. In the presence of incompleteness, this use case is slightly different from the previous one as the query can have different modalities. It can take the form of: does the user have the value X for feature Y? This makes sense if we have an application that offers support only under special conditions. But it could also take the form of: Is it possible (with a certain error probability) that a user has the value X for feature Y. This is useful for strategies where some features are highly critical (e.g. emotional state, but also social relationships), but evidence is usually rather scarce.
- **Query for other users in a certain context**. Especially in the area of corporate learning where the social dimension plays an important role, it is highly desirable to establish contacts between employees with a similar context. The system can

recommend others who are now or have been in a similar situation within a certain time frame in the past. This is a very different use case from the first one as it does not only cover the current context, but also previous contexts.
– **Trigger actions based on context changes**. Especially for proactive system behaviour, it is important that context-aware applications get a timely notification that the context has changed. One example are process or task changes that can initiate learning processes [7].

Typical context sources in this scenario either determine higher-level abstraction of the user's situation by analyzing user interface events (e.g. via Bayesian Networks[8] or rule-based formalisms), or by applying heuristics to data in existing sources like personal information managers (e.g. Microsoft Outlook) or documents.

2.3 Requirements

A closer inspection of the application scenario has revealed the following requirements:

– **Aging**. It should be obvious that collected context is not valid indefinitely. If the system gets to know about the current "task" of the user, this information will only be valid for a limited amount of time. As a consequence, the user context management system needs to have some aging mechanism.
– **Variability in dynamic behavior**. The closer inspection of the "aging" problem reveals that aging is not uniform across the different aspects of user context information. While information like name, birthdate changes infrequently to never, other aspects like personal skills, interests goals evolve over time, and tasks or location are highly volatile. So the aging support has to be specific for the different parts of the context.
– **Scalability**. If we want to materialize user context information, we have to select methods that are scalable with respect to large numbers of users and long time frames.
– **Push and pull interaction paradigm**. The system must support both the push and the pull paradigm, i.e. it must be able to answer interactive queries and allows for trigger-style notifications.
– **Open-world assumption**. As we have seen, the mapping between the situation is incomplete, and application may require to be able to query also for the possibility of facts or may exploit negative facts directly. The most trivial case is explicit negative user feedback on user context information, but there are many techniques that can generate negative facts in order to increase the quality of the context. So the context data model must build on the open world assumption.

3 Layered Context Model

3.1 General Considerations

For traditional database management systems, it has proven effective to divide the management functionality into different layers which are basically independent of the internal logic of the lower layers. In that spirit, we have grounded our work on a three layer

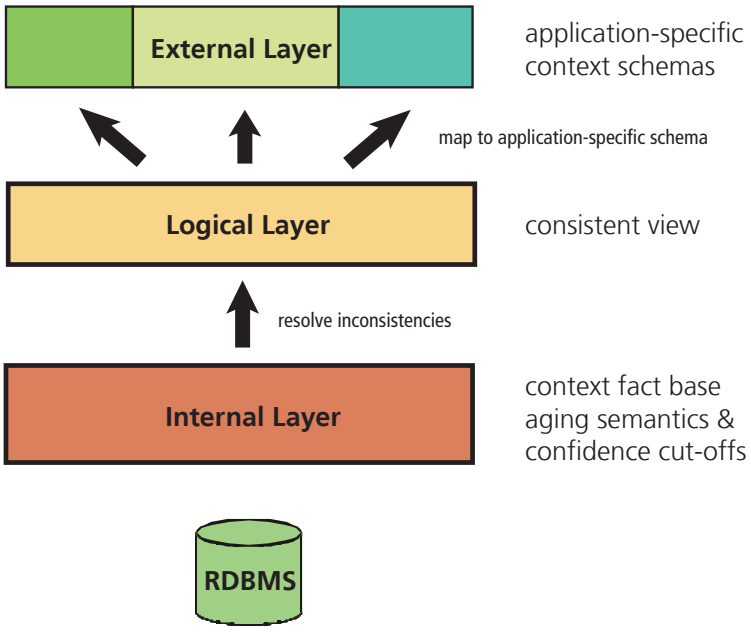

Fig. 2. Layers of the Context Data Model

model (an initial version of which has been presented in [9]) that allows for structuring the problem in a better way (see figure 1).

- **Internal Layer.** The internal layer as the lowermost layer stores all collected information about users in a time-dependent way as so-called context facts. The context facts can be queried according to their timestamp or by using value-level operators (on different data types).
- **Logical Layer.** This layer provides a consistent view on the collected data, conforming to a (single) specified schema. For an specific instant in time, the service on this layer can provide a consistent and semantically enriched view (based on schema-level information or background knowledge) on the context facts that adheres to certain quality criteria.
- **External Layer.** The top-most layer represents the usage context of a particular application at a certain instant of time. The context information is in the schema the application understands, which could be different from the logical schema.

These layers must not be confused with an aggregation hierarchy. How to achieve aggregation and abstraction from lower level information to higher level information is presented in section 3.7.

3.2 Internal Layer

The internal layer represents the lowest level of abstraction. It provides the basic functionality required for storing and accessing collected context information together with meta information about time, validity and confidence of the collected data. More formally, a context fact is defined as follows:

Definition 1 (context fact) *A context fact is a tupel* $(U, f, o, v, t, valid, \alpha)$ *where*

- U *is a user*
- f *is context feature*
- $o \in \{=, \neq\}$ *signals a positive or negative fact*
- v *is a value*
- $valid$ *is the validity interval for the value*
- t *is the point in time at which the factum was added to the fact base.*
- α *is the probability that at point of time* t *the feature* f *has the value* v *for user* U*.*

The set of all context facts is called context fact base and denoted with C

The support for negative context facts directly results from the requirement for an open-world assumption. In practice, this can be used, e.g., for explicit user feedback on certain infered facts about her context.

Definition 2 (Context feature) *A context feature is a tuple* $f = (uri, T)$ *where* (uri) *is a unique identifier,* $V = (T, O)$ *a data type consisting of a value space* V *and operators* O

The data type can represent traditional atomic data types like integers, strings etc., but also ontological data (the supported operators will be discussed in section 4), as the following examples show:*(Andreas, performs-task, =, literature-search, [2005-04-15 10:00,* ∞*), 2005-04-15 10:00, 0.8)*, and an entry *(Andreas, performs-task, =, examine-students, [2005-04-14 14:00,* ∞*), 2005-04-15 13:00, 0.9)*.

As additional schema-level information, the internal layer has **aging functions** attached to each context feature, which allow for describing how the confidence in a certain value decreases over time. An **aging function** basically is a monotonically decreasing function $a : TIME \rightarrow [0, 1]$, which is multiplied with the initial confidence value in order to obtain the current confidence value. These aging functions can be assigned heuristically or – preferably – based on empirical results. Queries from the logical layer can use the current confidence, which is calculated for each fact: If $c = (U, f, v, o, t, valid, \alpha)$ is a context fact, t^* the instant in time of interest. Then the confidence for c at t^* can be calculated as:

$$confidence(c, t^*) := [A(f)](t^* - t) \cdot \alpha$$

with $A(f)$ denoting the aging function associated with the context feature f.

3.3 Logical Model

The internal layer is rather ugly to use for context-aware applications. This has mainly to do with the fact that you can have almost arbitrarily inconsistent data. In analogy to traditional databases, we need the notion of a schema guarantee. A context schema is comprised of (1) a definition of context features (as above), (2) cardinality constraints and (3) a feature hierarchy.

Cardinality constraints are a very effective instrument for checking for an elementary check of consistency. For many features, it is clear from the application semantics that there can be only one value at a time so that any application also expects only a single value.

In order to allow for better reusing context information in different applications, the model also offers the possibility to define a *feature hierarchy* via feature inheritance, which directly corresponds to property hierarchies in RDF(S). This adds a basic inferencing capability to the model: if an applications requests the value(s) for a specific feature, the values of sub-features can be also returned. This can be formalized as follows:

Definition 3 (Feature hierarchy) *A **feature hierarchy** is an acyclical relation $H \subseteq (F \times F)$ on the set of context features F. Additionally, the following properties must hold for a feature hierarchy to be compatible with the feature set F:*

- $\forall(f_1, f_2) \in H$*: the value space of f_2 is a subset of the value space of f_1*
- $\forall(f_1, f_2) \in H$*: if f_2 is multi-valued then f_1 must also be multi-valued*

H^* is the transitive closure of H.

Based on this, the **context feature schema** C can be defined as $C = (F, card, H)$, where F is a set of context features, $card : F \rightarrow \{1, \infty\}$ is the cardinality assignment and H is a compatible feature hierarchy.

Definition 4 (Schema conformity) *A set $K = \{c_1, ..., c_n\}$ of context facts with $c_i = (U_i, f_i, v_i, \alpha_i)$ is conforming to a schema $C = (F, card, H)$ iff*

a) $\forall i \in \{1, ..., n\} : f_i \in F$ *and v_i is in the value space of f_i.*
b) With $values(f, U) := \{v | k = (U, f^*, ' =', v, \alpha) \in K, (f, f^*) \in H^*\}$ *the following must hold:* $\forall f \in F : |values(f)| \leq card(f)$
c) With $values^{\neg}(f, U) := \{v | k = (U, f^*, ' \neq', v, \alpha) \in K, (f, f^*) \in H^*\}$ *the following must hold:* $values(f, U) \cap values^{\neg}(f, U) = \emptyset$

This notion of schema conformity is essential for imposing a well-defined semantics ontop of imperfect data. It basically states that (a) we have only well-defined context features with associated data type definitions and that the facts conform to these data type constraints, (b) only multivalued features have multiple values, and (c) we have no contradictions resulting from positive and negative facts.

3.4 Mapping the Internal Layer to the Logical Layer

The main mapping task is the resolution of inconsistencies. Inconsistency occurs in our model if there are multiple values for a feature for which the cardinality constraints enforce a single value, or if we have positive and negative facts on the same feature. *Conflict resolution strategies* are responsible for transforming a set of context facts on the internal layer into a set of context facts conforming to the context schema. There can be different strategies to resolve these inconsistencies. The most obvious is to take the value with the highest confidence, but usually the strategy also needs to take into account that facts can be reinforced by other facts (e.g. two independent methods determine the same feature value within a limited time window).

If we apply this procedure to the example, it is clear that the restriction to a specific instant in time (e.g. *2005-04-14 11:00*) still provides two possible tasks. After applying the aging function, let's suppose that the *literature-search* has confidence 0.7 and *examine-students* has confidence 0.1. This would lead to a simple resolution strategy taking the *literature-search* as the current feature value, because we have specified that the *performs-task* feature is only single-valued.

As conflict resolution strategies, the maximum confidence strategy with some heuristic refinements has turned out to be quite effective for single-valued features. For multivalued context features, we are experimenting with strategies based on the Dempster-Shafer theory, which allows for aggregating probabilities from different sources [10].

Apart from conflict resolution, the mapping involves exploiting the feature hierarchy. This is done by rewriting a query for fact f^* to queries for the set of features $S_{f^*} = \{f \in F | (f, f^*) \in H^*\}$, i.e., all subfeatures.

3.5 The Problem of Asynchronous Notification

The most typical interaction pattern for context-awareness are publisher-subscriber patterns where the user context management service provides notifications about changes to the user context. Although we have apparently an append-only semantics on the internal layer (as typical for continuous query scenarios, [11]), there are two critical points resulting from the fact that the actual context is a time-dependent view:

- Aging makes the queries non-monotonous ([12]), i.e., it is not enough to provide additional results, but rather previous results have to be retracted. As a consequence, the notification protocol needs to incoporate both additions and removal of parts of the user context.
- Confidence-based filtering and conflict resolution are aggregation operators on the temporal data stream. These operators have to implemented in a way that they are – in most cases – self-maintainable so that with new arriving data, the changes to the view can be calculated without querying.

3.6 External Layer and Mapping from the Logical Layer

The external layer is intended to be interface for application, providing an application-specific view. In this step, the global context schema used on the logical layer is translated into an application-specific schema. This problem can be dealt with similar to

schema mapping techniques in classical information integration approaches. In case of simple projections and renamings, this can be done within the user context management system, but for more powerful mapping features, external mapping services are the method of choice (in spirit of [13]).

A far more challenging problem is when we consider mappings not only on the level of user context schemas, but rather also on the value level, especially in case of ontology-based data types (see below).

3.7 Resulting system architecture

From the model presented in this section, a high-level system architecture can be directly derived (see fig. 3.7). It shows the different layers at which the context management service can be accessed and sketches also typical added-value services and other components.

Below the internal level, two different types of context sources are supported: push and pull context sources. Push context sources notify the context management service of changes; their context information is materialized inside the system. Pull context sources can be queries on demand as soon as queries for covered features arrive. The reasoning service provides access to ontological data.

Added-value services are useful to improve the context quality or to facilitate the management task. Currently, we have two types of services: agents that use the user context management system as a blackboard and services that are used by the management system. The most important agents are Augmentation Agents (inferring additional facts from already collected ones) and Subcontext Agents (trying to exploit dependencies in the context to improve context switching and thus the time needed to adapt to changes). Augmentation agents can either work on the raw context facts, or on the consolidated view, depending on the inferencing methods: the logical layer will be preferred by logic-based approaches, whereas imperfection-aware methods will prefer the internal layer.

Many user modeling systems concentrate on the problem of abstracting and aggregating lower-level context information to information on a higher level of abstraction. Although this is not the focus of this architecture (which is on imperfection handling), this can be easily realized using a hierarchy of user context management services. Lower level services are plugged in as pull context sources into higher level services.

4 Integrating Ontologies

4.1 Motivation

Approaches to context modelling like [14] or [15] and to applying context awareness to the e-learning and similar domains like [16], [17], [18] emphasize the potential of applying Semantic Web technologies to user context management. It enables the creation of more semantically aware processing methods, especially by introducing a shared vocabulary, which can be used across different tools and systems, and by applying reasoning techniques based on domain knowledge.

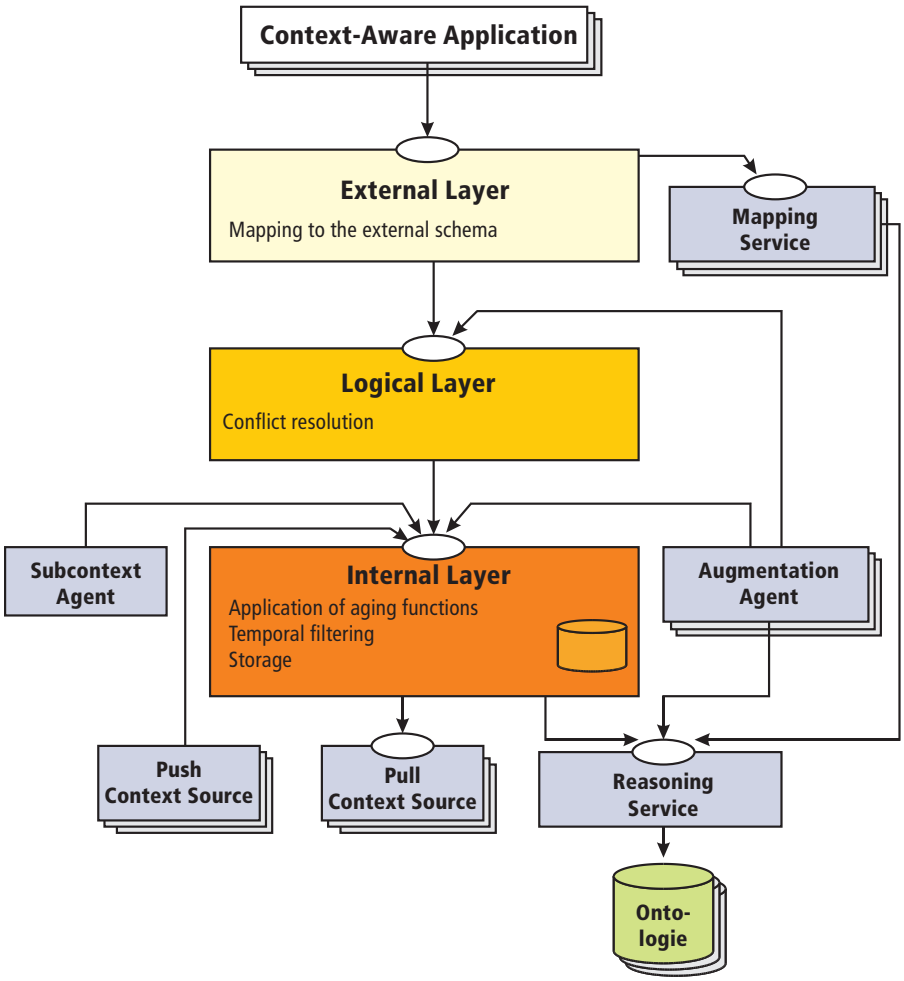

Fig. 3. High-level architecture of the user context management service

On the other side, Semantic Web technologies have still quite a way to go for solutions that are comparable in terms of scalability with traditional data management solutions. This is especially true for traditional types of queries like datatype specific range queries (e.g. for temporal data), although the description logics community tries to approach this problem with concrete domains (see e.g. [19]) and datalog-based reasoning (e.g. [20]). Also these techniques are not well-suited for highly dynamic scenarios with a high volume of changes as they so far do not consider update operations at all.

4.2 Approach

If we analyze our problem domain, it turns out that the benefits of ontologies are (beyond the ontology-like constructs like the feature hierarchy introduced in the last section) basically on the value level. We want to reference instances from the ontology in as feature values (e.g. the task in the examples above or a competency from a competency catalog). Also the inferencing capabilities (i.e. mainly classification) of descriptions logics are mainly needed for queries like (a) is the user in a task of the type X or (b) retrieve all users with expert-level competency in a certain subject area.

So the idea is to treat ontologies as a datatype that has certain predicates that can be used in a query to the user context management system, just like operators on dates ($\leq$, *between ... and* etc.) or numbers or strings. As there is currently no notion of operators for description logics in spirit of the operators of a data model and especially no standardized query language for (although there are some proposals like OWL-QL [21] or SPARQL [22]), we started with the most obvious operator *instance-of*, which already covers a large portions of practical cases encountered in our application domain. A sample query for users in *myDepartment* who were involved in a accounting process activity in August 2006 would be:

```
SELECT USERS
WHERE  process-activity instance-of AccountingActivity
       AND department = myDepartment
VALID  [2006-08-01,2006-08-31]
```

In the user context management service, this is handled by splitting the query into a query to the ontology service and a query to context fact base. Similar to join optimization, the processing order is determined based on result set estimations. In this case, a query to the ontology reasoner would give all instances of AccountingActitivity. These query would be rewritten as follows:

```
SELECT USERS
WHERE  (process-activity = a1
          OR process-activity = a2 ...)
       AND department = myDepartment
VALID  [2006-08-01,2006-08-31]
```

Alternatively, the query results of the user context fact base could be filtered with the help of the ontology, which is the preferred way if we have large sets of instances.

We are currently extending the approach with SPARQL subqueries to support more powerful navigation.

5 Implementation Case Study

The user context management infrastructure has been successfully used in the project *Learning in Process*, which was committed to implementing a learning support for context-steered learning, and further developments based on that system.

5.1 How the user context management infrastructure was implemented

The user context management service was implemented ontop of a relational database management system. The first prototype was based on PostgreSQL, but in subsequent versions we have also used Oracle 10g XE. For the internal layer, the most challenging issue was the efficient implementation of the calculation of the current confidence. The main problem is that we cannot rewrite queries involving the calculated attribute *current-confidence* to range queries on the fact attributes. In order to avoid calculation of We used a combination of techniques to speed it up, among them (a) using a subsuming query to prefilter the facts to calculate the confidence on by using approximations of the initial confidence, (b) historizing old facts and (c) using Oracle function-based indexes.

The implementation of the integration of ontologies is somewhat similar to approaches like [23] where ontologies are used for query rewriting, although the access of ontologies cannot be considered a preprocessing step. The query condition is split into parts that can be shipped to the underlying database system and parts that are shipped to the reasoner. Furthermore, this approach is basically a join between context facts and ontology results. As a consequence, we tried to optimize the execution order by making use of result size estimates, e.g. by using statistics on the number of instances (for the *instance-of* operator). As an ontology management system, KAON [24] was used in the first prototype of the system, using Java-API access. Currently we are moving towards KAON2 [25] for OWL-DL and SPARQL support and

5.2 How user context was used

As context features, a fairly broad range was used, which was divided into four categories: personal, social, organizational and technical. As personal context features, mainly learner preferences (semantic density, interactivity level) and competencies and interests were used. For characterizing the social context, we relied on a basic social relationship ontology. On the organizational level, we relied on organizational unit, role, business process activity and task. Technical features were user agent (browser, operating system, plugins), bandwidth, and availability of audio (input and output). Whereas the relationship between personal and technical context features and the available learning material was fairly straightforward, we needed additional background knowledge to relation organizational entities. This was mainly done by attaching competency requirements to organizational entities, which was encoded in an organizational ontology.

This context information was exploited by the so-called *Matching Service* which computes based on the background knowledge and the current user's context a competency gap and can compile personalized learning programs based on that gap that take into account the various aspects of of the user's context [5]. The Matching Service only

operates on-demand. In order to be able to realize proactive behaviour (i.e., recommending learning material to users based on context changes), the architecture additionally consisted of a *Learning Assistant*, which subscribes to the user context management service for context changes. Whereas the Matching Service decides on *what to deliver*, the Learning Assistant decides on *when to deliver* and displays the recommendations in an unobtrusive manner.

5.3 How user context was acquired

In contrast to prior work in the area of business-process-oriented knowledge management [26], we could not rely on a workflow management infrastructure to capture the organizational part of context. Rather, we experimented with a variety of heuristic sensors for application events, e.g., a plugin for Microsoft Office (relying on template information), a Browser Helper Object for Internet Explorer (for URL-based heuristics) and a plugin for Mozilla-based browsers. Additionally, we built interfaces to HR applications to extract the more static part of the organizational context.

For personal part of the context, we relied on static information from the user, but we are currently investigating the possibility of inferring the learner characteristics from application sensor data, e.g., by considering time of day, previous and upcoming meetings or other appointments etc.

For social relationships, we used mainly annotated address books as a (pull) context source, but similar to the personal level, we are currently investigating egocentric network analysis methods [27] to discover these relationships, e.g., from email conversations.

5.4 Results

After integrating the user context management service into the system, we conducted an evaluation with around 20 users at two companies. It has turned out that the system behaviour was perceived as useful by the evaluation participants.

On a more technical level, our tests have shown that the user context management service improves the robustness of the whole system in comparison to a naive approach in which we do not handle imperfection. Furthermore, the architecture has proven useful for plugging in and out different very loosely coupled context sources.

We are currently setting up a simulation environment for measuring the increase in quality and completeness with different assumptions about the context sources.

6 Related Work

6.1 Context modeling

There are plenty of models for dealing with user context information, both from the traditional community of user modeling and the recently emerged communities for context-awareness. A good overview of recent context modeling approaches gives [28]. In general, it can be stated that the data management problem is a neglected area of research. Most approaches either ignore the problems of imperfection and dynamics (e.g.

[29], [14], [15]), or assume that context can be accessed via the pull paradigm, which is certainly valid in sensor-based areas, but not appropriate for context information on a high level of abstraction. This can also be traced back to the fact that especially approaches to high-level context information rely on ontology-based techniques where imperfection is hard to integrate (although approaches exist, e.g. [30]).

6.2 Imperfection handling in context modeling

The consideration of the imperfection and dynamics of user context information is also a relatively neglected area of research, especially for the case of high-level context information. [31] investigate quality criteria for context information complementing quality of service concepts. They define the following criteria: precision, confidence, trust level (for context sources), granularity and up-to-dateness. [32] introduce meta attributes like precision, certainty, last update and update rate, the approach of [17] is similar. Only [33] has investigated the role of imperfection in a more systematic way and identified the following types of imperfection: unknown values, contradictory values, imprecise values, and incorrect values. Feature values are further classified according to their source and persistence into sensed, static, profiled and derived. The causes of imperfection are analyzed along this classification. But all of these approaches do not consider the implication on a management infrastructure.

6.3 Imperfection handling in data management in general

The major part of research on handling imperfection in data management seems to be almost a decade ago (see e.g. [34], [35] and [36] for an overview). Apart from fuzzy logic, the major part of research in data management concentrated on probalistic extensions of the relational and other data models. Two main approaches can be identified: probabilistic attributes ([37], [38], [39]) and probabilistic relations (i.e. probabilities on tupel level) [40], [41]. Current approaches are mainly concerned with semistructured XML data (e.g. [42], [43]). However, these approaches did not consider the time dimension (i.e. the problem of aging).

There are some combinations of temporal and imperfection problems, but these approaches concentrate on the imperfection of the temporal perspective itself (e.g. [44], [45]), not on the impact of time distance on the quality.

7 Conclusions and Outlook

The approach of this paper views robust and scalable user context management as a key enabler for rolling out context-aware application in the large. In order to retain robustness in the presence of imperfection, the system needs to consciously manage the imperfect properties of the data ([46]). This approach covers the uncertainty (via attached probabilities), the incompleteness (via open-world semantics), and contradictions (via storing contradictory facts and conflict resolution strategies). Additionally, the approachs also accounts for the dynamics of change of context information by introducing aging functions that decrease the certainty over time. These aging functions

are specific for context feature in order to deal with the variability in the rate of change between different aspects of the context. A layered approach helps to keep the complexity manageable and the architecture of the system extensible. Scalability is achieved through relying on traditional data management techniques and providing appropriate indexing structures for imperfection handling. Still it is possible to reference semantically rich ontologies as data types and accessing limited reasoning functionality within queries. The system has been successfully applied to a corporate learning scenario.

Central storage of user context data always raises (justified) privacy concerns. The presented architecture can be easily extended to support a distributed approach where context data is stored for each user separately, e.g., on her machine. There will be one single user context management service without any local storage that simply distributes (after checking the permission) the query to the individual context management systems, which are registered as pull context sources.

Future work will incorporate the research in a data type that captures imprecision via a probability distribution of that value, which is important for location information. A lot of previous research exists on that topic that can be integrated into the approach. Different conflict resolution strategies will also be evaluated with their effect on the context quality. For that purpose, agent-based simulation techniques will be used.

Acknowledgements

This work was partially supported by the European Commission under the Fifth Framework Programme of IST within the project *Learning in Process* (contract IST-2001-32518) and under the Sixth Framework Programme of IST within the project *AGENT-DYSL*.

References

1. Henricksen, K., Indulska, J.: Modeling and using imperfect context information. In: Second IEEE International Conference on Pervasive Computing and Communications. Workshop on Context Modelling and Reasoning (CoMoRea 04), IEEE Computer Society (2004) 33–37
2. Winograd, T.: Architectures for context. Human-Computer Interaction **16** (2001)
3. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In Mattern, F., Naghshineh, M., eds.: Pervasive 2002, Berlin, Springer (2002) 167–180
4. Schmidt, A.: Bridging the gap between knowledge management and e-learning with context-aware corporate learning solutions. In Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T., eds.: Professional Knowledge Management. Third Biennial Conference, WM 2005, Kaiserlautern, Germany, April 2005. Revised Selected Papers. Volume 3782 of Lecture Notes in Artificial Intelligence., Springer (2005) 203–213
5. Schmidt, A.: Context-steered learning: The Learning in Process approach. In: IEEE International Conference on Advanced Learning Technologies (ICALT '04), Joensuu, Finland, IEEE Computer Society (2004) 684–686
6. Schmidt, A.: Potentials and challenges of context awareness for learning solutions. In: LWA 2005: Lernen–Wissensentdeckung–Adaptivität, 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems (ABIS 2005), Saarbrücken. (2005)

7. Schmidt, A., Winterhalter, C.: User context aware delivery of e-learning material: Approach and architecture. Journal of Universal Computer Science (JUCS) **10** (2004) 28–36
8. Horvitz, E., Breese, J., Heckermann, D., Hovel, D., Rommelse, K.: The lumière project: Bayesian user modeling for inferring the goals and needs of software users. In: 14th International Conference on Uncertainty in Artificial Intelligence, Madison, Wisconsin (1998) 256–265
9. Schmidt, A.: A layered model for user context management with controlled aging and imperfection handling. In Roth-Berghofer, T.R., Schulz, S., Leake, D.B., eds.: Modeling and Retrieval of Context. Proceedings of the 2nd International Workshop on Modeling and Retrieval of Context MRC 2005, Edinburgh, Scotland, July 31 - August 1, 2005. Number 3946 in Lecture Notes in Artificial Intelligence (2006)
10. Ruthven, I., Lalmas, M.: Using dempster-shafer's theory of evidence to combine aspects of information use. Journal of Intelligent Systems **19** (2002) 267–302
11. Babu, S., Widom, J.: Continuous queries over data streams. SIGMOD Rec. **30** (2001) 109–120
12. Terry, D., Goldberg, D., Nichols, D., Oki, B.: Continuous queries over append-only databases. In: SIGMOD '92: Proceedings of the 1992 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (1992) 321–330
13. Kazakos, W., Nagypal, G., Schmidt, A., Tomczyk, P.: Xi3 - towards an integration web. In: 12th Workshop on Information Technology and Systems (WITS '02), Barcelona, Spain (2002)
14. Wang, X., Gu, T., Zhang, D., Pung, H.: Ontology based context modeling and reasoning using owl. In: IEEE International Conference on Pervasive Computing and Communication (PerCom'04), Orlando, Florida. (2004)
15. Strang, T., Linnhoff-Popien, C., Frank, K.: CoOL: A Context Ontology Language to enable Contextual Interoperability. In Stefani, J.B., Dameure, I., Hagimont, D., eds.: LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003). Volume 2893 of Lecture Notes in Computer Science (LNCS)., Paris/France, Springer Verlag (2003) 236–247
16. Nebel, I., Smith, B., Paschke, R.: A user profiling component with the aid of user ontologies. In: Workshop Learning - Teaching - Knowledge - Adaptivity (LLWA 03), Karlsruhe. (2003)
17. Heckmann, D.: A specialized representation for ubiquitous computing and user modeling. In: First Workshop on User Modeling for Ubiquitous Computing, UM 2003. (2003)
18. Dolog, P., Nejdl, W.: Challenges and benefits of the semantic web for user modelling. In: AH2003 Workshop at WWW2003. (2003)
19. Lutz, C.: Description logics with concrete domains - a survey. In Balbiani, P., Suzuki, N.Y., Wolter, F., Zakharyaschev, M., eds.: Advances in Modal Logics Volume 4. King's College Publications (2003)
20. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics with a concrete domain in the framework of resolution. In: Proc. of the 16th European Conference on Artificial Intelligence (ECAI 2004), August, 2004, Valencia, Spain. (2004) 353–357
21. Fikes, R., Hayes, P., Horrocks, I.: Owl-ql: A language for deductive query answering on the semantic web. Journal on Web Semantics **2** (2005)
22. Prud'hommeaux, E., Seaborne, A.: Sparql query language for rdf. W3C Working Draft 20. February 2006, W3C (2006)
23. Necib, C.B., Freytag, J.C.: Query processing using ontologies. In: Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAISE'05), Porto, Portugal. (2005)
24. Maedche, A., Motik, B., Stojanovic, L.: Managing multiple and distributed ontologies in the semantic web. VLDB Journal **12** (2003) 286–302

25. Hustadt, U., Motik, B., Sattler, U.: Reducing shiq-description logic to disjunctive datalog programs. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004. (2004) 152–162
26. Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O., Sintek, M.: Context-aware, proactive delivery of task-specific information: The knowmore project. DFKI GmbH International Journal on Information Systems Frontiers (ISF) **2** (2000) 139–162
27. Fisher, D.: Using egocentric networks to understand communication. IEEE Internet Computing **2005** (2005) 20–28
28. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England. (2004)
29. Meissen, U., Pfennigschmidt, S., Voisard, A., Wahnfried, T.: Context- and situation-awareness in information logistics. In Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., Vakali, A., eds.: Current Trends in Database Technology - EDBT 2004 Workshops, EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb,Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers. Volume 3268 of Lecture Notes in Computer Science., Springer (2004) 335–344
30. Nottelmann, H., Fuhr, N.: pdaml+oil: A probabilistic extension to daml+oil based on probabilistic datalog. In: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia, Italy (2004)
31. Buchholz, T., Küpper, A., Schiffers, M.: Quality of context: What it is and why we need it. In: 10th International Workshop of the HP OpenView University Association (HPOVUA 2003), Geneva, Switzerland. (2003)
32. Judd, G., Steenkiste, P.: Providing contextual information to ubiquitous computing applications. In: 1st IEEE Conference on Pervasive Computing and Communication (PerCom 03), Fort Worth. (2003) 133–142
33. Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. In: PerCom, IEEE Computer Society (2004) 77–86
34. Motro, A.: Management of uncertainty in database systems. In Kim, W., ed.: Modern Database Systems: the Object Model, Interoperability and Beyond. Addison-Wesley/ACM Press (1994) 457–476
35. Motro, A.: Sources of uncertainty, imprecision and inconsistency in information systems. In Motro, A., Smets, P., eds.: Uncertainty Management in Information Systems: From Needs to Solutions. Kluwer Academic Publishers (1996) 9–34
36. Parsons, S.: Current approaches to handling imperfect information in data and knowledge bases. IEEE Transactions on Knowledge and Data Engineering **8** (1996) 353–372
37. Barbará, D., García-Molina, H., Porter, D.: The Management of Probabilistic Data. ACM Transactions on Knowledge and Data Engineering **4** (1992) 487–502
38. Dey, D., Sarkar, S.: A probabilistic relational model and algebra. ACM Transactions on Database Systems **21** (1996) 339–369
39. Lakshmanan, L.V.S., Leone, N., Ross, R., Subrahmanian, V.: Probview: A flexible probabilistic database system. ACM Transactions on Database Systems **22** (1997) 419–469
40. Cavallo, R., Pittarelli, M.: The theory of probabilistic databases. In: VLDB '87: Proceedings of the 13th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1987) 71–81
41. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Transactions on Information Systems **15** (1997) 32–66
42. Nierman, A., Jagadish, H.V.: Protdb: Probabilistic data in xml. In: Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002. (2002)

43. Hung, E., Getoor, L., Subrahmanian, V.S.: Pxml: A probabilistic semistructured data model and algebra. In: Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India, IEEE Computer Society (2003) 467–
44. Dyreson, C., Snodgrass, R.: Supporting valid-time indeterminacy. ACM Transactions on Database Systems **23** (1998) 1–57
45. Dekhtyar, A., Ross, R., Subrahmanian, V.: Probabilistic temporal databases i: Algebra. ACM Transactions on Database Systems **26** (2001) 41–95
46. Lockemann, P.C., Lukacs, G.: Imperfection and the human component: Adding robustness to global information systems. In Brinkkemper, S., Lindencrona, E., Slvberg, A., eds.: Information Systems Engineering: State of the Art and Research Themes. Springer (2000) 3–14