

A Layered Model for User Context Management with Controlled Aging and Imperfection Handling

Andreas Schmidt

FZI Research Center for Information Technologies, Karlsruhe, Germany
Andreas.Schmidt@fzi.de

Abstract. Current research in context-awareness is biased toward low-level context information. High-level context information, however, poses several challenges to context management systems, which can be traced back to the asynchronicity of context acquisition and use and the inherent dynamics and imperfection in that process. This paper presents a three layer model allowing for dealing with the problems of imperfection and aging in a controlled way. It conceives the problem of high-level user context management as an information management problem with specific requirements.

1 Introduction

1.1 The Need for High-Level Context Information

With the ever-increasing volume of accessible information and the advent of ubiquitous information access via mobile and wearable devices, the focus of information systems research has shifted to increasing the efficiency of information access for the user. This encompasses both simplifying the query formulation process and improving the relevance of query results. In general, there is a trade-off between preciseness (and thus selectivity) of queries and ease of query formulation. The most promising approach is to incorporate information about the user and her current situation (i.e., her "context" [1]), which is the fundamental approach in context-aware and situation-aware systems.

The upsurge of interest in context-awareness has mainly occurred in the area of *low-level context information* as defined by [2], which represents information about the user's context that can be directly sensed (or obtained) or aggregated from this sensor data in a relatively straight forward way (although the concrete implementation may still pose severe problems). The most prominent examples here are location-aware systems. But recently, it has been discovered that the consideration of context is a key enabler for next-generation information services on a much broader scale. One example are e-learning and knowledge management systems ([3], [4]), which require rather high-level context information like tasks, business process steps, or the information whether a person currently is under time pressure or has some time to learn.

Apart from determining how the context influences the information need of a user, the key problem in these systems is how to keep the user context information up to date, which includes both the acquisition and the management of such information over time. In general, only indirect methods can be used for high-level context information, and several sources of context information have to be considered. Some information

can be derived from the user's interaction with a specific applications, other pieces can be obtained from data stored in other systems, e.g., in a corporate environment from Human Resources data, Workflow Management Systems [5], or Personal Information Managers (addresses, calendar etc.), still others can be retrieved on demand with specialized operations (e.g., localization in wireless networks). It has been realized (see also [6]) that this complexity should be hidden from a context-aware application by establishing a "user context management" middleware (or "broker architecture" [7]) that provides applications with an up-to-date view of the user's current context, partially materializing it, partially retrieving it dynamically from other sources.

1.2 Challenges for High-Level User Context Information Management

However, this idea faces several challenges, which cannot be adequately met by existing information management solutions [8]:

- **Dynamics.** The context of a user continuously changes. Different features of the context change at different pace; e.g., name and occupation change less frequently than location or current task. Additionally, we have to distinguish between context evolution and context switching. In the latter case, part of the context changes, but it is quite likely that it later changes back again so that we should store the information for later use. Typical examples are private and professional information, project-specific context and role-specific context.
- **Imperfection.** User context information is typically collected via indirect methods that observe a user's behavior and infer facts from these observations. These methods do not yield certain results, in some case they are more, in other cases less probable (uncertainty). Additionally, some of the information cannot be determined exactly (imprecision). The most typical example here is location information. Depending on the method (GSM, GPS, etc.), the precision of the information can vary a lot, which is particularly important for the most prominent examples of context-aware services: location-based services. Additional aspects of imperfection in the area of user context information are incompleteness and (as we collect it from several sources and/or with a variety of methods) inconsistency.

1.3 Overview

In the remaining part of this paper, a context information architecture is presented that is able to deal with these two challenges. First, the specific requirements posed by high-level context information to the respective management infrastructure are briefly discussed. Then a basic layered context information architecture is introduced that deals with the problems of aging and imperfection. In the following section, an extension with the concept of subcontexts is introduced that improves the handling of the dynamic nature of user context information. In section 4, there will be an overview of the prototypical implementation in the project "Learning in Process".

1.4 Specific Requirements for a Suitable Context Model

The context information architecture proposed in this paper focuses on the so far neglected issue of high-level context information as defined by [2]. This differs from mainstream context-awareness research because it poses special requirements to a context management infrastructure. The differences can be traced back to a main distinction: *Asynchronicity*. In contrast to low-level context information, high-level context information typically cannot be continuously monitored (via sensors) or determined on demand at any instant in time (e.g. GPS positioning). Rather – due to the required complex abstraction process –, the system has to collect in advance information about the user and make it persistent over time. This "materialized" approach introduces several problems (which can be ignored in low-level context settings):

- **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.
- **Reasoning over time**. The methods used to derive more abstract user context information typically do not solely rely on the current user's context, but also on the history in order to detect patterns.
- **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.

2 Layered Context Model

2.1 General Considerations

Current approaches to context modelling like [9] or [10] and to applying context awareness to the e-learning and similar domains like [11], [12], [13] 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. But Semantic Web technologies have still quite a way to go for solutions that are comparable in terms of scalability with traditional information management solutions. To take that into account, the context model presented here was developed as a data model (in the tradition of relational or object-oriented databases) whose core does not depend on those reasoning techniques, but which can be integrated smoothly with those techniques if a domain-specific schema requires it.

The second issue is directly related to the architecture of user context information management systems. 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 want to present a three layer model that allows for structuring the problem in a better way (see figure 1):

- **External Layer.** This layer represents the usage context of a particular application at a certain instant of time. The context information in the schema the application understands, and also certain quality criteria (like minimum confidence) are guaranteed.
- **Logical Layer.** This layer provides a complete view on the context of a user at a specific instant in time together with the imperfection metadata that allows for determining the reliability of the stored information.
- **Internal Layer.** The internal layer stores all collected information about users in a time-dependent way.

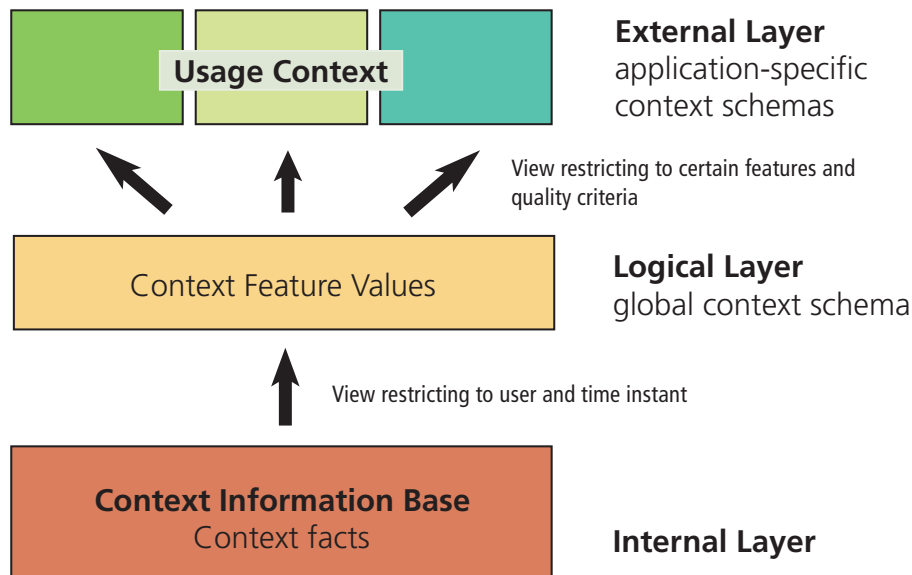


Fig. 1. Layered Context Model

In analogy to database systems, we begin with describing the logical layer (which corresponds, e.g., to the relational data model), then explain the internal layer and how the internal layer maps to the logical layer, before explaining the external layer (which corresponds to the concept of views) and how the mapping to this layer is realized.

2.2 Logical Model

The primary construct of the data model to describe the context of a user is the *context feature*, which corresponds to attributes (in case of the relational or object-oriented data model) or properties (in case of RDF(S) and other conceptual data models like OWL). Context features are described by a unique identifier (URI), a value space, and cardinality constraints (whether it is multi-valued or not). The value space can either consist of atomic data types (like numbers, dates etc.) or of concept or instance identifiers referencing elements from an ontology. That way, also inferencing on the value level is possible by reusing description logics based reasoners and ontology management systems.

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.

Feature values are tuples (U, f, v, α) , where U is the user, f is the feature, v is the value and α is the confidence level that the feature f currently has the value v for the user U .

An example would be $(Andreas, performs-task, literature-search, 0.8)$, which encodes that the user *Andreas* currently performs the tasks of *literature-search* with the probability of 0.8, which could have been determined from monitoring his usage of a web browser and the visited sites. *literature-search* could a reference to a concept in an ontology that allows for generalizing this concept to *search*.

The operations supported at this layer are

- *queries* for specific context feature values of a user and for users having certain feature values
- *update operations* that can set or delete feature values for a specific user

2.3 Internal Layer

In contrast to the logical layer, the internal layer does not only have to provide the current view on the user context, but it also needs to store the history so that temporal concepts need to be considered. The internal layer primarily consists of a fact base (called *context information base*) with context facts as its entries:

A *context fact* is a tuple $(U, v, t, valid, \alpha)$ where

- U is a user
- f is context feature
- 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 .

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 function $f : TIME \rightarrow$

[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.

Taking the example from above, the context information base could contain (*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*).

The operation on this layer are much more powerful as they can additionally exploit the temporal perspective.

2.4 Mapping the Internal Layer to the Logical Layer

In order to map from the internal layer to the logical layer, the following issues need to be taken care of:

- **Restrict to a specific point in time.** The set of context facts is restricted to those context facts for which the validity interval contains the requested instant of time.
- **Apply aging functions.** With the help of aging functions, the current confidence value needs to be calculated.
- **Infer additional information.** As the context facts only represent those values that were explicitly added to the fact base, we provide also a feature hierarchy on the logical layer, for which additional feature values need to be inferred.
- **Resolve inconsistencies.** Inconsistency occurs in our model if there are multiple values for a feature for which the cardinality constraints enforce a single value. 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 the resolution strategy to take the *literature-search* as the current feature value, because we have specified that the *performs-task* feature is only single-valued.

2.5 External Layer and Mapping from the Logical Layer

The external layer is intended to be interface for application, providing an application-specific view. On the one side, this consists of an application-specific context schema, on the other side the application can specify a certain quality-level, which depends on the usage strategy. This quality level is expressed as a minimum confidence level for supplied user context information. Where user context information is only used as a rough indication about the user, the confidence does not matter that much, but for applications that involve legally binding transactions, certainty about values is crucial.

In order to perform the mapping from the logical layer, the following two issues need to be taken care of:

- **Apply quality criteria.** This involves the filtering of the available feature values according to the supplied minimum confidence.
- **Perform a mapping.** In this step, the global context schema used on the logical layer is translated into an application-specific schema. 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 [14]).

3 Subcontext Switching Support

3.1 The Concept of Subcontexts

The layered context model introduced so far supports the controlled *forgetting* of outdated context information and allows for representing different levels of certainty. However, it does not help with the problem of slow adaptation to a changed context. This is due to the phenomenon that there are often dependencies between different features, i.e. groups of values often change together. A typical example are different roles (e.g. private role or business role). If a user currently has the private role, a wide range of context information can be different, e.g. his payment preferences, contact information etc. In order to cope with these dependencies, the concept of subcontexts is introduced. Subcontexts are basically groups of feature values that change together and have the following characteristics:

- Subcontexts can be nested.
- Subcontexts conform to a schema that defines (1) available features and (2) nesting relationships.
- For each subcontext schema, there is at most one sub context active at a certain instant of time. The others are present as inactive, "parallel" sub contexts yielding the same aspects about the user.

An example for a subcontext structure is given in figure 2. Here you have a user with location-dependent, role and project-dependent information. Currently the user "John Smith" is at his office and thus has broadband network access, but no loud-speakers (which could influence whether the system can deliver video or audio material). The system also knows the characteristics of other locations, but those are represented as currently inactive sub contexts. There is also role-specific information and within each role project-specific information. In the case of corporate e-learning, this information could in its majority be provided manually by a learning coordinator or human resources department.

In our three layer model, the concept of subcontexts is located at the internal layer.

3.2 Heuristic Strategies for Subcontext Management

Although it is possible and in many cases also reasonable to have the user manually indicate her current context, this limits the concept of subcontexts to a relatively coarse granularity and less frequent context switching. Ongoing research therefore tries to identify heuristic strategies how to automate the handling. The most crucial issues are:

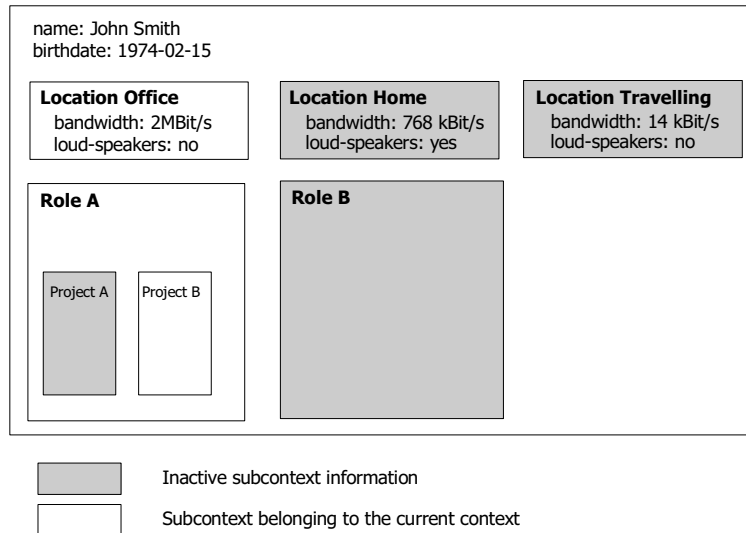


Fig. 2. Sample sub context structure

- **Detection of subcontext changes.** The most crucial part of the research is how to detect context changes, or to be more precise: how to distinguish context evolution from context switching.
- **Automatic construction of subcontexts.** Although being quite practical for closed environment like intranet e-learning solutions, the manual specification of schemas for subcontexts limits the scope of a generic user context management service. Therefore automatic methods for sub context detection are investigated. Promising approaches here are Data Mining approaches, but they have to be adapted to (1) the scarce amount of available data and (2) uncertainty aspects.

Currently, simple strategies have been implemented. For the detection of sub context changes, the strategy works with pivot features, which serve as semantic keys for sub contexts. If these features change, the rest of the feature value is also assumed to change. For the construction of subcontexts, a method based on functional dependencies is used (which is borrowed from schema normalization in relational database schemas. Further strategies are under research.

4 Implementation

A context management system based on the model presented in this paper was implemented in the project *Learning in Process*, which aimed at – among other things – supporting a new type of learning process for workplace learning: *context-steered learning*[15]. Instead of having human resource development experts assigning courses to employees or leaving it to the employees to actively search for learning resources

satisfying their knowledge need, the LIP system continuously monitors the employee's working activity (i.e. her context) and deduces from it (with the help of domain knowledge) the possible knowledge gap. Based on this gap, the system can recommend relevant learning resources, which can (but need not) initiate learning processes. The context schema used for that purpose incorporates the following:

- **Personal characteristics:** competencies, goals, learning preferences like interactivity level and semantic density
- **Organizational aspects:** role, organizational unit, business process (step), task
- **Technical aspects:** user agent, available bandwidth, available multimedia equipment

For more details on the context schema and how it was constructed see [4].

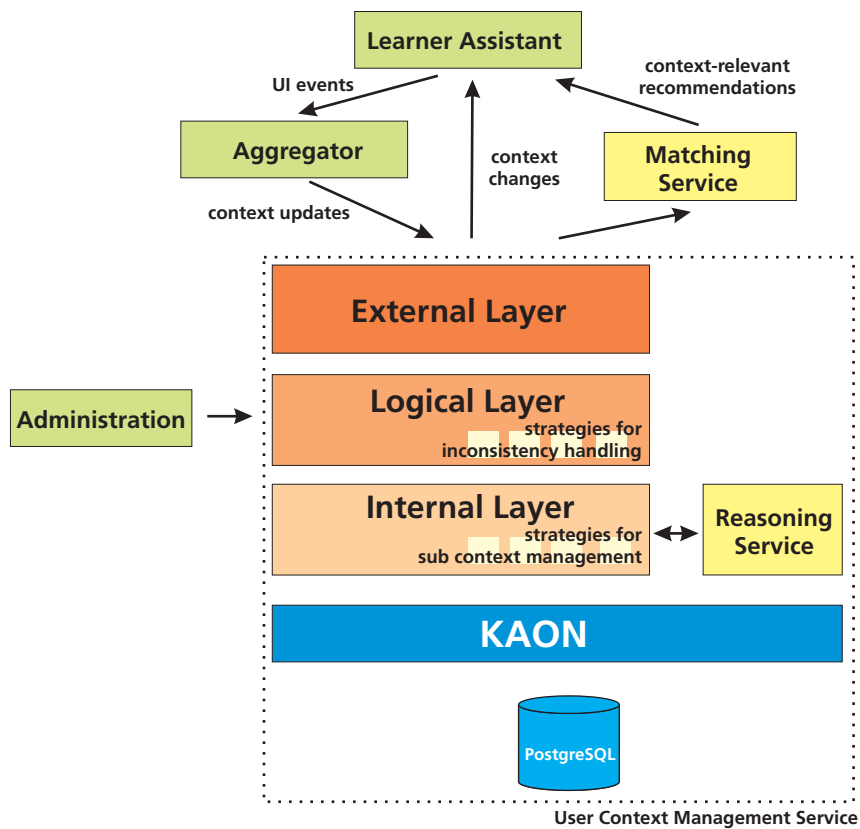


Fig. 3. Overview of the prototypical implementation in LIP

The basis for the service matching the current work situation and relevant learning resources is a user context management service that provides a view on the current context of the employee (see figure 2). In order to be able to easily implement feature inheritance and to smoothly work with moderately expressive ontologies used for encoding the domain knowledge, the implementation of the *internal layer* was based on the ontology management system KAON[16]. KAON supports a variation of the RDFS data model and is implemented on top of relational databases that support (at least some variation of) SQL-99. The operations of the internal layer were exposed both via specialized methods and via a descriptive query language (based on the KAON query language). For the project prototype, however, only the specialized methods were used by external applications.

Queries to the *logical layer* are also expressed in the KAON query language – mainly because it allows for smoother interoperability with the rest of the system. The user context features are mapped to virtual properties of the RDFS data model. The queries are rewritten to queries to the lower layer. Inconsistency resolution is done by postprocessing the results, but currently there is only a simple strategy in place (as described above). In the prototype, the *external layer* was very thin and did not require mapping services so far. As a consequence, there was only a confidence cut-off.

As *context sources*, on the one side there was a event-triggered update of more stable elements like name, role or organizational units. On the other side (*Administration* in figure 2), there were desktop *learning assistants* monitoring the interactions of the employee with her applications. For the two pilot installations, the following applications were observed: Internet Explorer, Microsoft Office (Excel, Word, Powerpoint), and Microsoft Visual Studio. The user interface events were aggregated and translated into context feature value changes.

In order to enable context-triggered actions, the service also supports notifications about context changes. These notifications were used by the learner assistant to determine when to invoke the *matching service* that retrieves relevant learning resources and compiles them into a personalized learning program based on the context of the user and background domain knowledge (for more details see [15] and [4]).

The simple strategies for inconsistency handling and subcontext switching were sufficient for the first prototypes. However, it is expected that with more context sources connected more sophisticated strategies will be required. The prototype provides a good basis for further experiments on this.

5 Related Work

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 [6]. In general, it can be stated that the data management problem is a neglected area of research.

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. [17] 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. [18] introduce meta-attributes like precision, certainty, last update and update rate. Only [19] 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.

6 Conclusions and Outlook

The proposed solution integrates the aspects of imperfection and dynamics into the context model and structures provides a layered structure similar to traditional data management applications. This allows for scalable solutions and appropriate decoupling of different management aspects. In addition to that it offers different interfaces to context-aware applications at different levels of complexity. Dedicated context sources or sophisticated context-aware services will access the context management infrastructure primarily on the internal layer, whereas applications that extend their traditional functionality with some context-aware features and provide the user only with limited interaction possibilities with the context information will primarily access the external layer. The logical layer is for added-value services that do not want to deal with the temporal perspective or inconsistent data.

Future research will explore the heuristic strategies used for inconsistency resolution and for subcontext switching via simulations. This will lead to insights which strategies are most suitable for specific domains characteristics.

References

1. Dey, A.K.: Understanding and using context. *Personal and Ubiquitous Computing Journal* **1** (2001) 4–7
2. Winograd, T.: Architectures for context. *Human-Computer Interaction* **16** (2001)
3. 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
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.: *Post Conference Proceedings of the 3rd Conference on Professional Knowledge Management - Experiences and Visions (WM05)*, Springer (2005)
5. Elst, L., Abecker, A., Maus, H.: Exploiting user and process context for knowledge management systems. In: *Workshop on User Modelling for Context-Aware Applications at UM 2001*. (2001)
6. 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)
7. Chen, H., Finin, T., Anupam, J.: Semantic web in the context broker architecture. In: *PerCom 2004*. (2004)

8. Schmidt, A.: Management of dynamic and imperfect user context information. In Meersman, R., Tari, Z., Corsaro, A., eds.: OTM Workshops. Volume 3292 of Lecture Notes in Computer Science., Springer (2004) 779–786
9. 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)
10. 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
11. 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)
12. Heckmann, D.: A specialized representation for ubiquitous computing and user modeling. In: First Workshop on User Modeling for Ubiquitous Computing, UM 2003. (2003)
13. Dolog, P., Nejdl, W.: Challenges and benefits of the semantic web for user modelling. In: AH2003 Workshop at WWW2003. (2003)
14. 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)
15. 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
16. Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz., R.: An infrastructure for searching, reusing and evolving distributed ontologies. In: Proceedings of WWW 2003, Budapest, Hungary. (2003)
17. 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)
18. Judd, G., Steenkiste, P.: Providing contextual information to ubiquitous computing applications. In: 1st IEEE Conference on Pervasive Computing and Communication (PerCom 03), Fort Worth, 133-142 (2003)
19. Henriksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. In: PerCom, IEEE Computer Society (2004) 77–86