Where do spatial context-models end and where do ontologies start? A proposal of a combined approach

Christian Becker
Distributed Systems
Universität Stuttgart
Universitätsstr. 38, 70569 Stuttgart
+49 711 7816 357
becker@informatik.uni-stuttgart.de

Daniela Nicklas
Applications of Parallel and Distributed Systems
Universität Stuttgart
Universitätsstr. 38, 70569 Stuttgart
+49 711 7816 217
nicklas@informatik.uni-stuttgart.de

ABSTRACT

Context-aware applications adapt their behavior depending on the state of the physical world along with other information representing context. This requires context management, i.e., the efficient management of context information and feasible context representations in order to allow reasoning. This paper discusses two common approaches, spatial context models and contextual ontologies, and argues for a combined approach providing the efficiency of context management through context models combined with the semantic expressiveness of ontologies.

Keywords

Context-Models, Adaptation, Infrastructures for Context-Aware Applications, Ubiquitous Computing

INTRODUCTION

Context-aware systems have attracted researchers in the past years starting from location-aware computing. Early work [1] considered context to be related to the location of users, the people nearby, and resources which can be accessed based on the spatial proximity. Depending on the focus of research projects, further definitions of context have been proposed. Projects related to Human Computer Interaction focused on user's activity or social environment, e.g., in order to adapt the behavior of a cell-phone [2]. The user's location was only of interest as long as it could be used to derive information about his activity.

The progress in technology with respect to the miniaturization of computing and sensing devices will lead to billions of information sources placed in our physical world which will constantly report changes in the physical world captured via sensors. This information is related to locations in the physical world as well as to users. This is an integral part of context concerning the locations as well as the users. Some existing information spaces also provide information about physical entities. Common to this information is the relation between physical entities (such as users or locations) and virtual entities (such as applications). This relation is independent of the origin of the information: it could be sensed by sensor platforms, provided by applications, by information spaces like WWW or geo-models from GIS.

Context-aware applications use context information in order to adapt their behavior. The different context sources and characteristics of context information, e.g., type and representation, has led to a number of different approaches to supply applications with context information. Besides specialized approaches, e.g., the context toolkit [3] for sensor integration or the location stack for positioning systems [4], two major classes of generic context management exist. *Context models* provide a database-style management of context information and typically offer interfaces for applications to query context information or receive notifications on context changes. *Contextual ontologies* address the need of applications to access a thorough representation of knowledge to reason about context information and to react accordingly.

This paper discusses the relation between spatial context models and ontologies in context management. A combined architecture is approached based on the classification of [5], and arguments are given with respect to the scalability and the appropriateness of abstractions for application programmers. Next, a brief classification of context and context-aware applications is presented. Context models are discussed in Section 3 and ontologies for context management in Section 4. A combined approach is presented in Section 5 before the paper closes with a conclusion and an outlook to further research.

CONTEXT AND CONTEXT AWARE APPLICATIONS

The following definition is based on the discussion in [6] and reflects a general view on context information, similar to the one in [7].

Definition Context: Context is the information which can be used to characterize the situation of an entity. Entities are persons, locations, or objects which are considered to be relevant for the behavior of an application. The entity itself is regarded as part of its context.

It is interesting to see that an entity can be part of its context itself as well as an entity can be interpreted different depending on the context.

For an example consider two applications dealing with trucks. A fleet management system would keep track about

the position, the freight, the route of trucks along with other information, such as the trucks' maintenance rate, the assigned driver etc. A navigation system installed in the individual trucks would also consider the delivery routes but only for the individual truck. Other information, such as traffic jam information, is used to optimize the navigation between destinations.

From a context management perspective both applications may operate on the same context information. The fleet management system may access the context model to retrieve all trucks in a given area in order to decide which truck a given tour should be assigned to. Individual trucks or drivers may be queried for administrative issues. The car navigation system accesses the context model to update information as well as to query for individual data relating the truck and its current position. The point in time is also crucial information for both applications. The navigation system will take the current situation into account. Prognoses of the traffic situation can be used to improve the route planning. Thus, the navigation system will access context data based on time (present and future). The fleet management system will also access information regarding the current time and the future for planning of tours. In addition to that, history plays an important role in order to account for transportation costs and for issuing invoices. Similar synergy can be found between different personal applications like tourist guides, personal navigation systems, augmented reality applications or smart environments.

Based on the scenario above we can derive that context information is accessed based on three major criteria:

- The identity of the entities
- The location of entities
- The time where the information is relevant

Because of the important role of identity, location, and time to the organisation of context models, we refer to these as primary context. The role of primary context in context management obviously is the indexing of context information. Further information of entities can be accessed once they are found using the primary index. The additional context information, e.g., load of a truck, a person's e-mail address, the vacancy of a taxi, are denoted as secondary context, even if they are more relevant in a distinct application domain. Possible combinations of accessing context information are *(location, time)*, *(identity, time)*, or *(identity, location, time)*. Note, that time may be used implicitly,

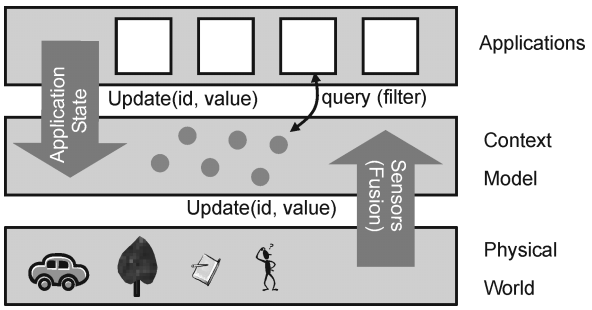

Figure 1: Context Model

e.g., an index may only refer to a distinct location. The time can then be interpreted as the current status of the context information related with the index.

The notion of primary and secondary context does not imply the relevance of context information from an application perspective.

Context-aware applications can make use of context in many ways. The following definition captures common understanding of context-aware applications [7, 6, 11].

Definition Context-Aware Application: An application is context-aware if it adapts its behavior depending on the context.

Note that in ontology-based systems, the term *agent* is used for both human users and software that interacts with the system. These will only differ in the way the context information is used and processed. An application, e.g., for navigation, will make use of positioning data in coordinates and derive information which are more suitable for users, such as a street name. The user can then choose further information, e.g., shops or restaurants, based on this information. If individual user profiles are stored, the system can provide a selection—based on this context information—as well.

Based on this definition, four classes of context-aware applications can be isolated, which either select information or services, change their presentation, or issue some action based on context, or tag information to context:

Context-based selection: Information and services which are used by an application are selected based on context information, such as a user's preference, their physical proximity (the next printer) or relevance to the user (public transport schedules from the next bus stop, touristic information, e.g. [12, 13]).

Context-based presentation: The way which and how information is presented to the user depends on the context. A navigation system may change the way information is displayed based on the speed of traveling from a map to a direction based output using arrows or to audio output only in order not to distract the user (e.g. [14]).

Context-based action: In contrast to context-aware presentation where a user is explicitly involved in the interaction with an application context-based action allows to automatically react to changes in the context without prompting the user. Examples are applications which automatically forward messages to the devices in a user's proximity, facility management systems which adjust light and heating conditions to user preferences (e.g. [15]).

Context-based tagging: In contrast to selection, presentation, and action, which lead to an immediate change in the behavior of an application, tagging of information to context allows a later action based on this information (e.g. Stick-E-Notes [16], GeoNotes [17], or Virtual Information Towers [13]).

CONTEXT MODELS

A general model for the relation between the physical world and context-aware applications based on context

models is depicted in Figure 1. The context model as shown in Figure 1 separates applications from the process of sensor processing and context fusion. Moreover, this allows a number of applications to share the gathered context. Note, that this is a conceptual model and depending on the underlying system, i.e., based on an infrastructure or ad hoc network, and the way applications make use of the context information the instances of this model may differ.

For example, context can be managed by applications without providing means for sharing. A number of such applications will manage their local context models. Other applications may not build an explicit context model but directly access sensor information and process the obtained context information directly. In the remaining part of this section we assume context information to be available via appropriate representations in a context model and discuss access to context information, the spatial organization, and classify context models.

Context access

Queries to a context model should support the selection of entities based on primary context as depicted in Figure 2. Depending on the application requirements, not all query types have to be supported. If access to context information of the past of future is not an issue, the context models only store the current state reflecting the present time—or the time where the context model has been captured—based on state from the physical world.

The service models supported should not only allow for queries in the pull-model but also allow for asynchronous communication. Spatial events [15] are an example where applications are notified about changes in the context and receive a notification. A spatial event is defined by a predicate which operates on context data. This allows to raise actions based on changes in the context model which are typically triggered by state changes in the physical world.

Spatial organization

One important aspect of context information is related to location. This includes the position of entities as well as the spatial relation to other entities. Such relations cover the inclusion in a distinct area or range and the distance to other entities. Typical queries a context management platform should support with respect to location are according to [18]:

Position: Retrieve the position of an object. Examples are *Where is John*?, *What is the position of printer PHP13*?.

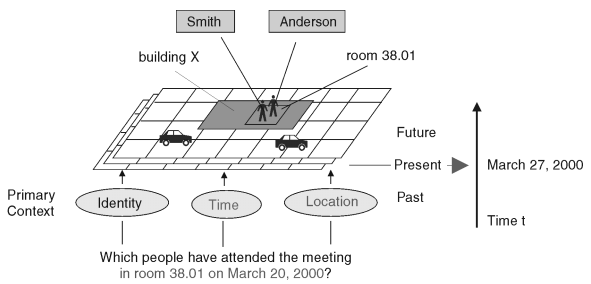

Figure 2: Queries to a context model

Range: A number of objects which are located in a spatial range are retrieved. Examples are *What objects are on Floor 2 of the Computer Science Faculty Building?* which includes all objects in the rooms on the second floor as well.

Nearest Neighbor: These queries offer a list of one or more objects which are closest to the position of an object. Queries for the next printer, restaurant, or gas station thus become possible.

Although these queries at first seem simple and obviously necessary for a variety of context-aware applications, their efficient processing depends on the underlying spatial structure and the involved coordinates by the position information. Position information is obtained by positioning systems which track mobile objects and report their position to a location management system. In general, two kinds of coordinates are supported by positioning systems:

Geometric coordinates: Represent points or areas in a metric space, such as WGS 84 coordinates of GPS which represent the latitude, longitude, and elevation above sea level of mobile objects. Using geometric functions such as the Euclidian distance allows calculating distances and allows for nearest neighbor queries. Overlaps of geometric figures can be used to specify ranges by their geometric extension and determine whether ranges are included in each other which allows for range queries.

Symbolic coordinates: Symbolic coordinates are represented by an identifier, such as a room number or the ID of a cell or access point in wireless telephone or local area networks. In contrast to geometric coordinates there is no spatial relation offered by symbolic coordinates. In order to allow spatial reasoning about inclusion (for ranges) and distances (for nearest neighbors) explicit information about the spatial relations between pairs of symbolic coordinates has to be provided. Note that this location model also is applicable if there is no explicit modeling of space but only by relations between objects (as in [19])

Location models are used to define spatial relations between locations. In general, locations can be determined by a symbolic identifier but also by a geometrically defined location. The latter allows expressing spatial relations which are not covered by the underlying metric on geometric coordinates. Choosing a suitable location model for the spatial structure of a context model is important for two reasons. First, the possible spatial queries along the primary context location depend on the location model. Secondly, the integration of two or more context models has to provide a mapping from one location model into another. This allows spatial queries across the objects with possibly different location information provided by different positioning system as basic coordinates or different spatial relationships modeled in graphs or hierarchies.

Context model classification

Context models and their corresponding management architectures can be classified ([6] or Figure 3) along the dimensions of:

Spatial scope: Denotes the spatial area which is covered by the context model. This area can range from rooms in a smart environment over smart homes to global scope.

Complexity of abstractions: Refers to the level of detail and the details which are provided by the context model. Complex model could incorporate highly detailed 3D models of buildings whereas simple 2D models are commonly used, e.g., in navigation systems.

Dynamism: The rate in which updates to information in the context model is supported typically depends on the provided complexity and scope. Cell-phone networks allow for high dynamism with respect to the managed position of mobile users but rely on a rather simple context model representing the position of users in terms of the cells their mobile terminal is logged in.

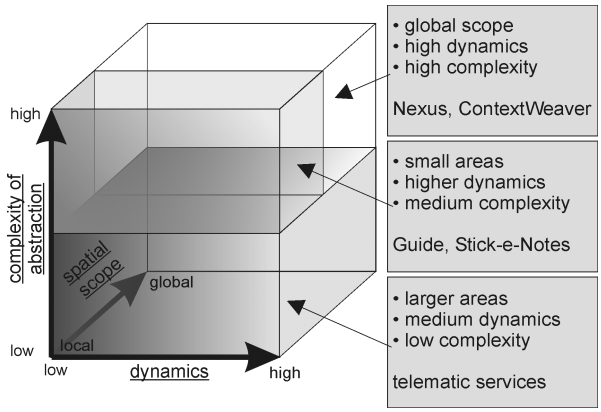

Figure 3: Classification of context models

These properties indicate that different kinds of context management are possible depending on the complexity of context information, the spatial scope, and the dynamism in which changes occur. Scalability of context management typically leads to a restriction in one of the properties. There are only few projects so far addressing context management for large scale, highly complex, and highly dynamic context information, e.g. Nexus [20] and ContextWeaver [22].

The interpretation of context information stored in a context model requires a common understanding, especially, if the context information is used by more than one information and may be supplied by third parties.

Context models differ in the way semantics of the context information is represented. Some offer explicit notions of context types, e.g., the GUIDE project [12] uses augmented HTML documents carrying the information about objects, Nexus uses an extensible class schema allowing for multiple inheritance [21]. Other are configured towards the needs of an application or application domain and do not provide explicit information about the context type. For instance, the context toolkit [3] configures the sensors and the fusion elements in an application specific way and does not foster context sharing. Applications making use of context information have to know about the type and interpretation of the context data.

Common to both approaches—implicit and explicit semantic represenation—is that the context types are typically obtained from domain knowledge, e.g. use case analyses. The representation of context follows the requirements of the data management. Typically, choosing benign representations for further context processing, like fusion, aggregation, or in general reasoning, is not an issue for these systems. This brings us to the next section where ontologies are discussed with respect to their use and relation to context-aware computing.

CONTEXTUAL ONTOLOGIES

There are already some approaches that support or implement context-aware applications by ontologies. They consider ontologies as a key requirement for building context-aware applications: they enable knowledge sharing, reasoning about contextual information and therefor interoperability between applications.

The Context Broker architecture (CoBrA) proposed in [8] is based on the ontology COBRA-ONT. It incorporates classes about physical places, agents (both humans and software), the location context of agents and the activity of agents. CoBrA focuses on scenarios where people on a university campus come together in a meeting and it needs 41 OWL classes and 36 properties for modeling its knowledge base.

CONON [10] is another context ontology for context-aware applications. It provides an upper context ontology that captures general concepts about basic context, and also provides extensibility for adding domain-specific ontology in a hierachical manner. In a prototype, a specific ontology for a smart home application domain was implemented using contains 197 OWL classes.

Andrew U. Frank proposes in [5] 5 tiers of ontologies that we consider very useful to discuss the relationship between context models and ontologies (see table 1).

Table 1: The five tiers of ontology according to [5]

Ontology Tier 0: Physical Reality *reality :: world –> property –> spacePoint –> timePoint –> value*
Ontology Tier 1: Observable Reality *observation :: world –> observationType –> location –> value*
Ontology Tier 2: Object World *observation :: id –> time –> observationType –> value*
Ontology Tier 3: Social Reality *getname :: object –> name* *findObject :: name –> environment –> object*
Ontology Tier 4: Cognitive Agents *rules used or deduction*

Tier 0 is the ontology of the physical reality. It contains the assumption that there is exactly one real world; hence, for every property in the world and for a given point in time-space there is a single value.

Tier 1 includes observations of reality. This is the first tier that can be accessed in computational systems. Here, a value can be derived at a location with a given observation type. The type determines the measurement scale of the value (e.g. nominal or rational) and the measurement unit (e.g. meters or seconds). For spatial values, also a coordi-

nate systems must be given. Values normally have a limited accuracy due to observation errors.

In tier 2, single observations are grouped together to individual objects that are defined by uniform properties. Now, the value of an observation is the state of a whole object, given by an id. Andrew U. Frank only considers physical objects in this tier, i.e. "things which exist in the physical world and can be observed by observation methods“. They have a geometric boundary in the world, but it can change over time (e.g. dunes or fluids).

Until now, the ontology tiers cover data that can be seen as *objective reality*—you can send out a team of geographers or students to model physical objects and they will come to an agreement about their observations. In tier 3, the socially constructed reality is represented. Social reality includes all the objects and relations which are created by social interactions. These are properties that are classified and named within the context of administrative, legal or institutional rules. Object names belong to this tier since they are assigned by culture; for many important things (but not all) there are functions to determine the name and to find the object by name in a certain environment.

Finally, in tier 4 the rules are modeled that are used by cognitive agents (both human and software) for deduction. This tier is normally built into database query languages, applications or inference engines of knowledge based systems.

DISCUSSION

In this section we discuss the basic properties of context models and ontologies. Based on some facts and derived hypotheses a combined approach integrating context models and ontologies is proposed.

Facts and Hypotheses

First, let us present some basic facts about context models and ontologies.

Fact 1: Context models can be large. Road networks for car navigation—although rather static—provide scalability for a complete country up to global scale. Other examples are location management of mobile terminals in cellular networks, Nexus [20] and ContextWeaver [22] (Figure 3).

Fact 2: Knowledge bases of contextual ontologies are rather small—compared to context models. The CoBrA and CONON ontologies presented in the last section serve as an example where a rather restricted model is used to reflect the semantic reasoning for a specific environment. There is—to the best of our knowledge—no global or large scale approach for ontology-based context management.

Fact 3: Sharing of context information is imperative. The example of the road network shows that current information—which obviously is expensive to be obtained—can be made available at little cost due to the economy of scale. Clearly, this forces applications to rely on common standards and representations in order to reuse the information. Also, management platforms have to deal with a high number of users and requests especially in case of distributed and dynamic information—in contrast to a navigation DVD which is local and static. Based on these facts we can derive two hypotheses:

Thesis 1: Ontologies provide sophisticated concepts for knowledge representation and reasoning. This is a tier 4 feature. The scalable management of context information, i.e., especially on tier 0-3, however, is not a core feature of ontology-based context management.

Thesis 2: Context models allow scalable context management. The knowledge representation of context models is typically rather straight-forward and does not provide suitable means for reasoning concepts.

A combined approach

Our proposed architecture combines the strengths of both approaches while trying not to carry the specific weaknesses into the resulting architecture. Context models clearly are well suited to query for information regarding a distinct spatial area and related objects. Figure 4 shows our proposed architecture where the tiers of ontologies are mapped to layers in the management architecture.

Context models are responsible to integrate context information obtained from different sources, e.g., sensors, geo-information systems, etc. into local context models (tier 1). To facilitate the sharing of context models a federation of these local context models has to be established. Clearly, this requires a common standard of context representation (tier 2, 3) as well as adequate query languages. This architecture so far looks like some approaches already existing in context aware computing. The layers above the federation tier can make use of the context information, e.g., an application directly interfaces to the context information. In order to allow for reasoning based on context information an integration of the context information into a suitable ontology is required.

The key idea represented in Figure 4 is the projection of the context information used by the reasoning layer into a global and scalable context management system. The context models residing below the reasoning layer provide the necessary information for a distinct reasoning task and thus reduce the complexity of the context information used in the reasoning process. The reasoning layer requests the required context information for a distinct reasoning task from the underlying federation layer. The reasoning layer is formed by either applications or a distinct inference machine which can be reused by a number of applications.

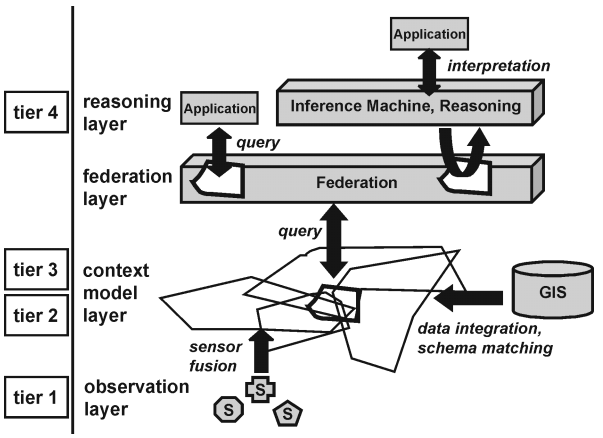

Figure 4: A combined approach

Similar to the federation which provides means for reuse on tier 3 an inference machine enables reuse on tier 4.

The current activities in ontology-based context management and context models are promising w.r.t. their specific strengths. The combination of both approaches requires a clear understanding of the responsibilities of tier 3 and 4 and their representation in the corresponding layers in our proposed architecture.

CONCLUSION AND OUTLOOK

Efficient context management and context representations that support for reasoning over context have been investigated separately so far. An analysis of ontology tiers and their corresponding responsibilities showed that a combined approach is possible mitigating the weaknesses of the single approaches. Context models allow to provide context information efficiently to reasoning based on ontologies. In order to realize such an approach we will investigate transitions between both context representations—context models and contextual ontologies.

ACKNOWLEDGMENTS

The Nexus project was funded 1999-2002 by the German Research Association (DFG) under the grant 200989 and is continued as Collaborative Research Center (SFB) 627 since 2003.

REFERENCES

1. B. N. Schilit, N. A., R. Want: Context-Aware Computing Applications. IEEE Workshop on Mobile Computing Systems and Applications, 1994
2. A. Schmidt, M. Beigl, and H.-W. Gellersen: There is more to context than location. Computers and Graphics 23(6), 1999
3. D. Salber, A. Dey, G. Abowd, G.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. Proc. of CHI 1999
4. J. Hightower, B. Brumitt, and G. Borriello: The Location Stack: A Layered Model for Location in Ubiquitous Computing, Proc. of the 4th IEEE Workshop on Mobile Computing Systems & Applications, 2002
5. A. U. Frank: Ontology for Spatio-Temporal Databases. M. Koubarakis et al. (Ed): Spatio-Temporal Databases—The CHOROCHRONOS Approach. Lecture Notes in Computer Science, Springer 2003
6. K. Rothermel, M. Bauer, C. Becker: Digitale Weltmodelle – Grundlage kontextbezogener Systeme. Total Vernetzt, Ed. F. Mattern, Springer, 2003 (in German)
7. A. Dey, G. Abowd: Towards a better understanding of context and context-awareness. Georgia Tech GVU Technical Report, GIT-GVU-99-22, 1999
8. H. Chen, T. Finin, and A. Joshi: An Ontology for Context-Aware Pervasive Computing Environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, Cambridge University Press, May 31, 2004
9. CYC Upper Ontology http://www.cyc.com/cycdoc/vocab/vocab-toc.html
10. X. H. Wang, D. Q. Zhang, T. Gu, H. K. Pung: Ontology Based Context Modeling and Reasoning using OWL. Proc. of 2nd IEEE Annual Conf. on Pervasive Computing and Communications, Workshop Context Modeling and Reasoning, IEEE Computer Society, 2004
11. G. Chen, D. Kotz: A Survey of Context-Aware Mobile Computing Research. Dartmouth Computer Science Technical Report TR2000-381, Dartmouth College, 2000
12. K. Cheverst, N. Davies, K. Mitchell, and A. Friday: Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. Proc. of the 6th Annual Intl. Conf. on Mobile Computing and Networking, Boston, Massachusetts, 2000
13. A. Leonhardi, U. Kubach, K. Rothermel: Virtual Information Towers—A metaphor for intuitive, location-aware information access in a mobile environment. Proc. of third Intl. Symposium on Wearable Computers, 1999
14. J. Baus, A. Krüger, W. Wahlster: A resource-adaptive mobile navigation system. Proc. of Intl. Conf. on Intelligent User Interfaces, San Francisco, 2002
15. M. Bauer, K. Rothermel: How to Observe Real-World Events through a Distributed World Model. to appear in: Proc. of the Tenth Intl. Conf. on Parallel and Distributed Systems, 2004
16. J. Pascoe: The Stick-e Note Architecture: Extending the Interface Beyond the User. Proc. of the Intl. Conf. on Intelligent User Interfaces. Editors, Moore, J., Edmonds, E., and Puerta, A., pp. 261-264, 1997
17. F. Espinoza, P. Person, A. Sandin, H. Nyström, E. Cacciatore, M. Bylund: GeoNotes: Social and Navigational Aspects. Proc. of UBICOMP 2001, Atlanta, USA, 2001
18. C. Becker, F. Dürr: On Location Models for Ubiquitous Computing. Accepted for publication in Personal and Ubiquitous Computing, Springer, 2004
19. D. A. Randell, A. G. Cohn: Modelling Topological and Metrical Properties in Physical Processes. Proc. of the First Intl. Conf. on the Principles of Knowledge Representation and Reasoning Processes, 1989
20. F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, M. Schwehm: Next Century Challenges: Nexus—An Open Global Infrastructure for Spatial-Aware Applications. Proc. of the Fifth Annual ACM/IEEE Intl. Conf. on Mobile Computing and Networking, 1999
21. D. Nicklas, B. Mitschang: On building location aware applications using an open platform based on the NEXUS Augmented World Model. In: Software and Systems Modeling, 2004
22. N. H. Cohen, A. Purakayastha, L. Wong, D. L. Yeh: iQueue: a pervasive data-composition framework. 3rd Intl. Conf. on Mobile Data Management, 2002