A Modular Data Infrastructure for Location-Based Services

Shijun Yu and Stefano Spaccapietra

Database Laboratory, I&C, EPFL, Switzerland
`shijun.yu@epfl.ch, stefano.spaccapietra@epfl.ch`

Abstract. Knowledgable Location-based Services (LBS) aim at enabling mobile users to specify their requests and profiles on the move and providing them with context-aware and personalized local information relevant to their current activity and request. Therefore, it opens up new challenges in data infrastructure, knowledge representations and data management etc. In this paper, firstly we will discuss the characteristics of LBS in terms of data management, and then present our data architecture. Finally, we will explain how the knowledge is incrementally set up and maintained in a modular manner.

1 Introduction and Motivation

Traditional data management applications operate in well-structured information environments and benefit from full-fledged strategies (e.g. SQL) that provide the needed functionality to represent, manage and query well-structured data. Web-based application environments, on the other hand, face a cumbersome task trying to provide the same functionality for the huge amount of heterogeneous data resources that reside on the Web. This new framework entails an emphasis on elicitation of data semantics, to improve the chances for correct interpretation of the data by heterogeneous partners (users and agents) that do not adhere a priori to common coordinated behavior. In parallel, users' increased mobility has lead to the development of Location-Based Services (LBS), i.e. services tailored to provide information that is selected taking into account the current location of the user on the move. Current LBS rely on traditional data management techniques, integrating all the data they need into a single centralized repository and providing all their users with the same services. We foresee that a new generation of LBS, which we call knowledgeable LBS, will be developed to provide enhanced services, namely contextualized and personalized information retrieval, and constant evolution to acquire new knowledge as available and as requested by users. Given the commonality between knowledgeable LBS and semantic web services in terms of facing an unbounded information space, LBS will likely use semantic web technology, enabling them to be used not only by users on the move, but also via the web, and exchange data and services with any agent within the semantic web. LBS services will nevertheless retain their specificities, namely:

- **Locality, Mobility and Dynamics.** LBS web services will provide information about a specific region, in particular information related with the current real or virtual location of the user. To fulfill this capability, LBS have to build and maintain knowledge repositories describing all locally relevant data stored in a limited number of data sources. When the user moves from her/his current place to a remote one (e.g. from Paris to London), the previous sources and knowledge become useless for upcoming queries from the user. Therefore, LBS need the capacity to dynamically acquire new sources and build the corresponding new knowledge into its repositories. Alternatively, LBS can be specialized to only serve a specific region. In this case, it will be up to users to switch from one LBS to another (just like cellphone users today switch from one telecommunication provider to another one). This paper gives some hints on how LBS may become knowledgeable about a specific region, which we see as a basic functionality for 2nd generation LBS.
- **Trading comprehensiveness for rapidity.** LBS intend to serve people on the move, i.e. they have to provide rapid rather than comprehensive responses to user queries. Well-known centralized management strategies that call for long set up processes and static solutions are not well suited to LBS needs. This departs significantly from e.g. data warehousing frameworks [4] that have similar data heterogeneity problems but different data quality and comprehensiveness requirements. Suitable query processing strategies need to be developed.
- **Modularity.** LBS obviously have a strong specific focus on spatial and temporal information and related constraints. In addition to space and time challenges, LBS aiming at context-awareness have to be highly sensitive to the current state of affairs when responding to user requests. Similarly, they have to care about personalizing services based on knowledge they can acquire about user's characteristics. This multiplicity and diversity of concerns make LBS a complex software whose processes constantly need to adjust to running circumstances. To make this possible while keeping performance we propose a modular data architecture, more easily manageable than the centralized approach in current LBS.

2 Related Work

Many different strategies can be applied in a LBS to organize, acquire and maintain the heterogeneous data involved in LBS. Earlier LBS were mainly concerned about a specific application, for instance, Active-Badge [9] to obtain the latest target's location, EasyLiving [7] to *'find a colleague in the building'*. In these systems, LBS infrastructures were relatively simple and mostly oriented towards to a single application supported by a central database. From a practitioners' viewpoint, a middleware-based architecture [1] is well-suited for LBS design due to its application-independence, adaptability to multiple platforms and dynamicity. A positioning service, for example, is an obvious component of a LBS middleware

model [3]. Similar proposals are also embodied in [2] and commercial products at IBM, ORACLE and Microsoft. In CRUMPET project [6], client-mediator-server three-tier approach was employed to improve the context-aware interactions between the user and services.

Crucial components for knowledgeable LBS are those supporting semantic interoperability and data reusability. As we have learnt from semantic web research, knowledge sharing and reasoning capabilities call for ontology management services. LBS need an ontological approach to solve typical issues about concept classification, knowledge inference, and concept similarity evaluation, just to name a few. We therefore propose a knowledge infrastructure relying on a domain ontology, alike e.g. a tourism ontology. In line with a web services view, we see the LBS ontology as an ontology of services and service usability. Because of this service orientation, the LBS ontology is to be equipped with specific features, such as links between services to show functional equivalence used to plan alternative services. Beyond service descriptions, our LBS ontology includes relevant contextual information that is essential to refine service usability given the current state of the local world. Similarly, it includes knowledge about user characterization used for personalization purposes. This makes up three knowledge modules that together define what we call the modular core ontology [5] as shown in Figure 1. Different from most ontologies, the LBS ontology has to have the generic knowledge and know-how to deal with space and time aspects and reasoning. For example, it has to know about points, lines and areas and about topological constraints. To emphasize this, Figure 1 shows two knowledge containers about space and time included in the core ontology. However, they are drawn differently because of the difference in nature wrt the user, context and service modules. Notice that Figure 1 only shows the repositories in the infrastructure, not the corresponding process components (cf. Section 6) used to build and maintain the repositories. Complementing the core ontology with rich terminological knowledge is important to support the variety of cultural and linguistic habits of LBS partners, through flexibility of vocabulary in the exchanges between the LBS and its partners (users, service providers, knowledge providers). The framework for these exchanges is provided by the mappings (cf. Figure 1) between LBS knowledge and the specific descriptions of the current LBS partners. We denote the latter as user/service/context profiles.

This paper presents and discusses the knowledge architecture that organizes the various components mentioned above, and how this knowledge is incrementally set up and maintained.

3   Our LBS's Data Infrastructure

By definition, a LBS holds an integrated view of data and services from sources local to a specific region (e.g. a city and its suburbs). Occasional extension of the geographic coverage can be achieved by considering the LBS as a peer in a peer-to-peer architecture [10]. Whenever the LBS is unable to answer a user query because the local data does not lead to a possible answer, before replying

negatively it may forward the query to geographically neighboring LBSs. For
instance, let us assume that user Shirley, currently in Lausanne, has to attend
a meeting in Geneva later in the day. She asks for time-tables of relevant buses
in Geneva. The LBS serving Lausanne, aware it is unable to answer her query,
may forward the query to the neighboring LBS in Geneva. This paper does not
address such typical P2P issues, it focuses instead on semantic data management
for a single LBS.

Regarding data management for a single LBS, we do not propose to inte-
grate or align all data from the sources into a single repository. Instead, LBS
knowledge only includes an abstract view of the source data. Thus, for example,
the detailed description of a service as stated by the service provider remains
at the source, while the abstract view maintained in the LBS only records the
main characteristics of the service, i.e. the minimal information that is needed
to quickly estimate if, given a specific query, the service may be relevant or not
for the querying user. Whenever more detailed information about a service is
needed, the sources are directly queried to extract the instantiations and other
properties of services that may be of interest to the requesting user. This ap-
proach shows two benefits: firstly, sources can autonomously maintain their data,
while the LBS is just responsible for suggesting users where to find appropriate
services. Secondly, the sources can protect their data based on their own privacy
regulations and can constrain users' access so as to only provide their data if
certain conditions are satisfied.

In order to build a knowledgeable LBS system capable of actually relating con-
cepts from different sources, disambiguating the terms in queries, and supporting
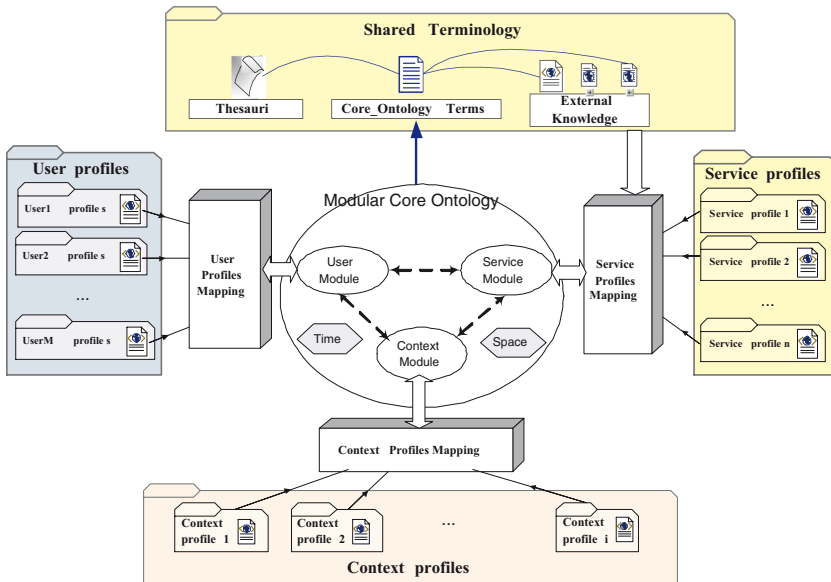

Fig. 1. The Basic Data Infrastructure in our LBS

query-service matching according to multiple criteria, so as to efficiently respond to user queries, the data infrastructure we propose includes six repositories interconnected via corresponding mappings (see Fig. 1). Two repositories, called Core Ontology and Shared Terminology, contain the domain-relevant knowledge built by the LBS and LBS administrators, and its related terms. Three other repositories contain knowledge on the external actors with whom the LBS interacts, i.e. users, services, and context providers. We say these repositories respectively contain the user profiles, the service profiles and the context profiles. Finally, the last repository (not shown in Fig. 1) is a working repository that contains the queries being processed by the LBS, thus holding data specific to a given query, and possibly the historical record of this data. For term similarity, we refer to this query data as query profiles. The infrastructure components are briefly described hereinafter to give an overall but intelligible view of the proposed infrastructure.

4   The Modular Core Ontology

The core ontology (CO) is the repository for the semantics of the data managed by the LBS. It is the kernel of the LBS's data infrastructure and it is used to structure the diverse aspects of service-related information and perform reasoning about it. It basically describes taxonomies of interest, encompassing a set of definitions of classes, properties, relations, axioms and constraints. These taxonomies organize information in complementary sub-domains. Services, users, context, space and time are the main sub-domain we have identified as essential to LBS. These sub-domains are quite heterogeneous and each one can be pretty complex in itself. Therefore, for better management and improved performance, we propose that the core ontology be a modular ontology composed by one module per sub-domain. A module is defined as a smaller ontology, which covers a sub-domain within the domain of the larger ontology. This fits perfectly with the LBS core ontology and its multiple taxonomies. Moreover, building modular ontologies is nowadays feasible. Several proposals exist on how to build a modular ontology, how to maintain modules individually as well as maintain the inter-module links that allow interactions between modules, and how to perform distributed reasoning within a modular organization [5]. Modular ontologies have been claimed to solve scalability issues and speed up reasoning, benefits that are for sure also relevant for LBS. However, the primary benefit we see in a modular ontology is its improved understandability and easiness of design and administration. A modular organization allows autonomous development of each module by an administrator with specific expertise in the sub-domain. Moreover, considering the semantic web and its world of specialized services, the development of modular approaches can improve the ontology's reusability. If good sub-domain repositories are available, it is much easier to reuse them than to elaborate a brand new one from scratch.

Whether modular or not, reuse is anyway the approach to follow when starting building the core ontology. Initializing the core ontology means inserting all

concepts that are assumed to be useful for the targeted application(s). Once the targeted knowledge domain identified, a clever designer looks for existing ontologies in the same or similar domain. If any one is found, its import can form the initial set-up for the core ontology. For example, if a tourist-support application is targeted, the designer will look for, and find, a tourism ontology whose concepts include accommodation, food, transport, and leisure services. Very likely, many of the imported concepts will be generic enough to be suitable for the new LBS and its service module. However, not all of them will be relevant for local use, and not all of them will be formulated in a way that is consistent with local habits. Therefore, in a second step, the core ontology is turned into a *local* domain ontology (e.g. tourism support in the Canton de Vaud, Switzerland). This can be done by acquiring and adding location-specific information (i.e. contextual data), such as local landmarks and local calendars, and enriching existing information, such as adding the preconditions for using a given available service, e.g. to use a motor-boat rental service the user needs to hold a valid sailing licence. Conversely, making the ontology local also includes removing generic concepts that are locally irrelevant (e.g. downhill skiing for an LBS about Amsterdam).

At this point, the LBS is fully ready to start operation. As long as the LBS is in use, the core ontology is expanded based on the queries received and answers given. Identification of new requirements by the ontology manager will also lead to ontology expansion, but this is very much similar to normal ontology evolution, not specific to LBS, and will not be discussed here. Notice that for the administration of the core ontology given this incremental strategy it is advisable that elements in the ontology be qualified as either prospective or confirmed. A prospective element is one that has been entered in the initialization phase but has not yet been used (up to now). A confirmed element is one that has been actually used during the processing of at least one query. Prospective elements should sooner or later become confirmed elements. Elements that remain prospective elements for too long are candidate for deletion, to be triggered by the ontology managers whenever it is felt that this is a reasonable enhancement to ontology performance (smaller ontologies may be explored and updated faster than large ontologies). More sophisticated maintenance strategies may be defined, based on usage metrics, relevance feedback and other usability criteria. They are not investigated here.

5   The Shared Terminology

In our proposal, the set of terms hold by the shared terminology is a superset of the terms in the core ontology. This superset is intended to provide extended terminological support to facilitate interoperability among components of the data infrastructure. It helps in solving heterogeneity issues (from the syntactic level to semantic level) such as differences in the vocabulary of data sources, core ontology and user queries, and related ambiguities. A typical example is the use of synonyms, e.g. a user querying for soccer matches while the ontology records

information on football matches. The shared terminology is queried by the LBS whenever the core ontology cannot identify a concept used in a query. For instance, a multi-lingual shared terminology may be used to overcome language limitations of a LBS based on a monolingual core ontology. An English-based LBS may thus be able to answer queries from French-speaking user.

We propose to initially populate the shared terminology with the concepts in the core ontology, which we call *internal terms*. Next, each internal term is complemented with its definition(s) extracted from certain thesauri. Subsequently, the shared terminology is further enriched by introducing what we call *external terms*, i.e. terms (from the thesauri) that are somehow relevant to the internal terms, but are absent from the core ontology. *Relevant* means there is a relationship between the internal term and the imported external term. For instance, the internal term 'car rental' may be related to the external term 'hire a car' with a semantic equivalence (synonymy) relationship. During LBS operation any new term appearing in user queries or in service-profiles will be identified thanks to the shared terminology manager and added to it in order to capture the terminologies of users and services that differ from the terminology of the LBS designers. Because the shared terminology manager can use any external ontology to identify the unknown term, we do not expect the identification process to fail. Should this happen, the query or service description using the term cannot be accepted and further human interaction is needed to solve the issue.

6   Information Profiling and Semantics Matching

Service Profiles. In our proposal, the descriptions of specific services are autonomously created by service providers and are kept and maintained at the corresponding data sources, external to the LBS. These service descriptions, together with some metadata about the data source (e.g. owner name, last update date), form what we call a service profile (see detailed definitions in [11]). Service profiles are acquired by the LBS when the data source joins the LBS. The acquisition process includes

Service Profile Matcher. This process component, the SP matcher, is responsible for acquiring the service profiles, recognizing the terms and concepts in the service descriptions, evaluate their matching with the knowledge in the service module of the core ontology, and accordingly establish the mappings between service profiles and service module. This mapping is used at query time to retrieve the services relevant to users' queries. The matching process uses a set of pre-defined syntactic and semantic rules to transform the heterogeneous service profiles into the format consistent with the LBS core ontology. The matching also produces the mapping of the core ontology into service descriptions, i.e. the syntactic and semantic support needed to transform users' queries into the format that the data sources can understand. The process is repeated for the classes, properties, and other features that the service profile encodes. The goal is not to fully copy the service profile descriptions into the ontology, but to ensure that the core ontology service module holds enough information about the

service to be able to evaluate the effectiveness of the service as a response to users' queries.

User Profiles. The LBS needs to know about its users to achieve its personalization goal. Descriptions of users, mostly in terms of ¡attribute, value¿ pairs about e.g. age, profession, preferences and dislikes, are traditionally called user profiles. We assume these profiles are stored in a dedicated repository that may be within the LBS or in a site elsewhere located but accessible by the LBS. Same as for service profiles, user profiles need to be understood by the LBS, i.e. the LBS must identify what the data in the user profile means and how it can be related to context and service data. The LBS user module maintains the concepts generically related to users, possibly abstracted from the actual user profiles.

User Profiles Matcher. This process component, the UP matcher, plays for user profiles management the same role and functionality the SP matcher plays for service profiles. A specificity of the UP matcher is to extract from the user profile information to guide the presentation of the query results to the user.

Context Profiles. The LBS needs to know what information participates into the description of the local framework (e.g., local events and places worth recording, local cultural habits such as shops opening hours), and the current status of the local world (e.g., local traffic conditions and whether the current day is a working day or a holiday). The LBS also needs to know existing correlations between context data and user/service data. This information identifies e.g. user preferences and service accessibility that are context-dependent.. These specifications form the context module, while factual context data is dynamically acquired from given data sources, e.g. weather data is extracted from the web pages of the local meteorological station. For each data source, we call context profiles the description of the locally available context data. They may contain, for example, a set of URLs with associated description of which kind of data is available and how it can be extracted, or a pointer to a tourist office database or set of XML files such as an event schedule.

Context Profiles Matcher. Similarly to the other matchers, this one is responsible for establishing and maintaining the mapping between the context module and the context profiles.

Query Profiles and Query Relaxation Profiles. The components in the basic infrastructure are those needed to understand user queries, refine it using contextual and user knowledge, and identify services that may match user's interest. Additional functionality may be achieved in terms of smarter query processing and calls for additional information that we define hereinafter as query profiles and query relaxation profiles (not shown in Figure 1). A query profile holds for each user query a description of its successive reformulations computed by the multi-step query processing strategy, as well as the relevant subsets of the user profile (conveying the user data that has been found relevant for this specific query) and of the context data (conveying the contextual data that has been found relevant for this specific query). Query profiles are

stored within the LBS as element of a sequence of queries that we makes up a user interaction, i.e. an exchange between the user and the LBS that leads the user, through a series of questions & answers, to get the desired information. In addition, query relaxation is a very common topic in LBS. Whenever the user needs additional results or a perfect matching can not be accomplished, a query relaxation strategy is activated. The strategy calls for user specifications about what can be relaxed (i.e. which selection criteria can be softened to a larger selection) and which ordering of relaxations is to be preferred. We call query relaxation profile this user-driven and query-specific specification. The analysis of query profiles can assist to refine the relaxation rules and ranking functions for better recommendation and more efficient query relaxation.

7   Interactions between Components

This section briefly shows a usage scenario to illustrate the interplay of the different components in the data infrastructure. We first show a set-up scenario, followed by a query scenario.

Set-up Scenario. Let us assume a software company has an LBS skeleton, i.e. all the software to run the planned location-based services, and wants to set-up a first version of its LBS servicing tourists visiting the city of Lausanne. Setting up this specific LBS is a knowledge acquisition process. As we already mentioned, the first task is to find and import one of the existing ontologies for the tourism domain. The imported concepts form a first draft for the service module, generically describing standard services in support of traveling tourists. The second initialization step is acquisition of the local context. This can be achieved through import of data files acquired from local providers (e.g. the local tourism-related organizations such as cultural associations, local press, movie distributors, transport companies, and so on). Alternatively, data can be captured from public websites using knowledge extraction techniques (e.g. [8]). Captured and acquired data are formatted to define and populate the context module. Such a priori knowledge of context may be needed to perform the following step, which is acquisition of service descriptions from local service providers. As stated, we assume that local providers will make their service descriptions available, not necessarily following a fixed format or adhering to a fixed terminology. This is where the LBS will start building the shared terminology, in its attempt to understand what a given service description means. For example, retrieving the service description term from WordNet and associating it to the corresponding term already in the service module. Acquiring service knowledge is a sophisticated task, as the goal is not to import service descriptions but to build the abstract view of services that forms the service module. This knowledge abstraction step relies on linguistic techniques (to identify major relevant terms) as well as on semantic techniques (to only retain what is useful in differentiating the service from the other services). Building the service module obviously includes building the mapping between the module and the service profiles. Once the service and context modules are set up, the LBS is ready to start receiving

queries from users. Here the question is whether user profiles will come from the user device or will have to be retrieved from some external repository of user profiles. Given the sensitivity of the issue, the former is likely to prevail. This means the LBS has to run a user profile understanding process, equivalent to the service profile understanding process. However, some generic knowledge about users characteristics can be initialized a priori within the user module. Actual user profiles will be matched against this initial set-up, the matching possibly leading to enriching the user module and the shared terminology.

Query Scenario. Here we just sketch what happens in the proposed LBS. User queries in our approach are formulated by stating which kind of service the user is interested in, and the spatial (e.g. proximity) and temporal (e.g. current availability) conditions that are to be taken into account while selecting specific services in response to the query. The user query can also specify thematic (non-spatial and non-temporal) conditions to refine the search for services of the given kind. The hypothetical query "Give me nearby restaurants still serving Swiss traditional cuisine after 2pm today" denotes restaurant as the desired a kind of service, together with the specification of spatial, temporal, and thematic conditions. Query processing within the LBS includes the following phases: 1) understanding the query (are the terms and the service they denote known in the core ontology or in the shared terminology? If not, search external ontologies); 2) retrieving from the service module the prototypical description of this kind of services in order to check preconditions for using services of this kind and check that conditions stated in the query can be actually evaluated (e.g. that the type of cuisine offered by a restaurant is known); 3) personalizing the query, i.e. check if relevant knowledge in the user profile allows refining the query, e.g. refining a query for a restaurant into a query for a restaurant within the price-range defined in user's preferences; 4) contextualizing the query, i.e. check if context data allows further refinement, e.g. replacing the qualitative expression "cheap restaurant" with the quantitative expression "restaurant with average price less than 40 CHF"; 5) executing the query, i.e. find relevant services in the order of preference, if any, as stated in the reformulated query. Notice that the order in which to execute phases 2, 3, and 4 can be changed without influencing the final result.

8   Conclusion and Future Work

In this paper, we first discussed the features characterizing data management in a LBS, i.e. locality, mobility, dynamics, heterogeneity, on the fly interoperability, and modularity. We emphasized the split in data organization between knowledge that is elaborated by the LBS itself (the core ontology and the shared terminology) and knowledge that is provided by external partners (users, context, and service providers). The core ontology represents the domain-specific conceptual views in a unified fashion. In addition, the shared terminology provides terminological support to overcome the heterogeneity problem that often

results from the diversity of the cultures and languages, as well as from the lack of the knowledge of the LBS' structure and naming for users.

Our modular data infrastructure is mainly based on OWL (W3C recommended Web Ontology Language). Its full-fledged support to processing queries and reasoning over a large scale of ontology repositories still calls upon the convergence of many efforts from both the semantic web community and industrial practitioners. We can envision many potential challenges in deploying the future LBS, e.g. defining a robust ontology query language, developing more powerful ontology management tools, enhancing reasoning capabilities on complex data types such as spatio-temporal data etc.

References

1. Bernstein, P.A.: Middleware: a model for distributed system services. Communications of the ACM 39(2), 86–98 (1996)
2. Cole, A., Duri, S., Munson, J., Murdock, J., Wood, D.: Adaptive Service Binding Middleware to Support Mobility. In: Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW 2003) (2003)
3. Jacobsen, H.-A.: Middleware for Location-Based Services. In: Schiller, J., Voisard, A. (eds.) Chapter of the book Location-Based Services, Morgan Kaufmann Publisher, San Francisco (2004)
4. Jensen, C., Friis-Christensen, A., Pedersen, T., Pfoser, D., Saltenis, S., Tryfona, N.: Location-Based Services - A Database Perspective. In: Proceedings of the Eighth Scandinavian Research Conference on Geographical Information Science, pp. 59–68 (2001)
5. Parent, C., Spaccapietra, S., Stuckenschmidt, H. (eds.): Ontology Modularization. Springer, Heidelberg (to appear, 2008)
6. Schmidt-Belz, B., Laamanen, H., Poslad, S., Zipf, A.: Location-based Mobile Tourist Services - First User Experiences. In: Proceedings of International Conference for Information and Communication Technologies in Travel and Tourism (ENTER), Helsinki, Finland (2003)
7. Shafer, S., Krumm, J., Brumitt, B., Meyers, B., Czerwinski, M., Robbins, D.: The New Easyliving Project at Microsoft Research. In: Proceedings of the Joint DARPA/NIST Smart Spaces Workshop (1998)
8. Tezuka, T., Lee, R., Kambayashi, Y., Takakura, H.: Web-Based Inference Rules for Processing Conceptual Geographical Relationships. In: Proceedings of the 2th International Conference on Web Information Systems Engineering WISE (2001)
9. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. ACM Transaction on Information Systems 10(1), 91–102 (1992)
10. Yu, S., Spaccapietra, S., Cullot, N., Aufaure, M.-A.: User Profiles in Location-based Services: Make Humans More Nomadic and Personalized. In: Proceedings of IASTED International Conference on Databases and Applications, Innsbruck, Austria (2004)
11. Yu, S.: Contextualized and Personalized Location-based Services. Thesis No. 3896, EPFL, Switzerland (2008)