Context Modelling and Context Awareness: steps forward in the Context-ADDICT project*

Cristiana Bolchini[1] and Giorgio Orsi[2] and Elisa Quintarelli[1] and Fabio A. Schreiber[1] and Letizia Tanca[1]
[1]Dipartimento di Elettronica e Infomazione - Politecnico di Milano
Piazza Leonardo da Vinci, 32 - 20133 Milano (Italy)
[2]Department of Computer Science - University of Oxford
Wolfson Building, Parks Road OX1 3QD Oxford (United Kingdom)
{bolchini,quintare,schreibe,tanca}@elet.polimi.it, giorgio.orsi@cs.ox.ac.uk

Abstract

We give an account of the researches on context-aware information tailoring which are going on within the PEDiGREE[1] group at Politecnico di Milano, starting from a foundational framework for the life-cycle of context-aware information systems, in which the system design and management activities consider context as an orthogonal, first-class citizen. The design-time and run-time activities involved in this life-cycle provide material for stimulating research, summarized in this paper.

1 Introduction

In a world of global networking, the increasing amount of heterogeneous information, available through a variety of channels, has made it difficult for users to find the right information at the right time and at the right level of detail. This is true not only when accessing information from portable, mobile devices, characterized by limited – although growing – resources and by high connection costs, but also when using powerful systems, since the amount of "out-of-context" answers to a given user request may be overwhelming. Therefore, we evolve from the need of personalizing information presentation to the difficult task of filtering and personalizing the information itself.

User tastes and profile, external environmental factors, current trends and involved phenomena are today recognized as parts of the notion of context [9, 30, 1, 6], enriching the initial concept only based on time and location. In the *cognitive-science view*, "context is used to model interactions and situations in a world of infinite breadth, and human behaviour is key in extracting a model"; in our work we focus on the less ambitious *engineering view*, where "context is useful for representing and reasoning about a restricted state space within which a problem can be solved" [9], and use it as the basis for information personalization and filtering.

Copyright 2011 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This research has been partially funded by the European Commission, Programme IDEASERC, Project 227977-SMScom. Giorgio Orsi is supported by the Oxford Martin School - Institute for the Future of Computing.

[1]PErvasive Database GRoup of EnginEers

Modern applications adopt a context-aware perspective to manage: i) *communication* among users and among systems [11], or between the system and the user [10, 14]; ii) *situation-awareness*, like modelling location and environment aspects (physical situation) [25, 22] or the current user activity (personal situation) [18]; iii) *knowledge chunks*: determining the set of situation-relevant information [27, 7], services [26, 16] or behaviours [4, 31].

Because the applications are so different, we feel that context modelling should be addressed independently of the specific objective, therefore on the side of the operational system we design the *context management system*, where all the above-mentioned variables contributing to context, rather that being considered just as all the other system parameters (*holistic* view), have the special role of *context variables* and are modelled orthogonally with respect to the other instantaneous system inputs. In the Context-ADDICT project [7] context is hierarchically modelled in terms of *observable* parameters that have a symbolic internal representation within a *context schema*[2], and some of which correspond to numerical values gathered from the environment by means of suitable appliances like sensors or RFID tags. At run time, the context is "sensed", and then validated, when the discovered combinations of values constituting the current context are verified against the context schema. This triggers a context-aware behaviour: delivery of context-aware data or actuation of context-aware operations.

This general view of the design-time and run-time activities involved in context-awareness encompasses all kinds of context-aware systems, however in Context-ADDICT we concentrate on a data-oriented perspective, where users are presented with the right information at the right moment, while all the data that are not really interesting for the current context are considered as "noise" and thus removed. This process is referred to as *context-aware information tailoring*.

The activities involved in context-aware information tailoring provide material for stimulating research, summarized with the help of Figure 1, which conceptually illustrates the vision adopted in Context-ADDICT to support all the phases of the tailoring process. We consider a general scenario where several classes of users access, through a (set of) application(s), a variety of information, made available by internal or external data sources, from sensors used for monitoring the environment to datasets of any format. The knowledge of the context is used to control the flow of information reaching the users, excluding the *information noise*. A fundamental aspect of the whole context management architecture is to keep the elaborations, queries and data retrieval operations necessary to present the context-tailored information transparent to the user, independently of the type of information source. The core of the architecture consists of the following main components:

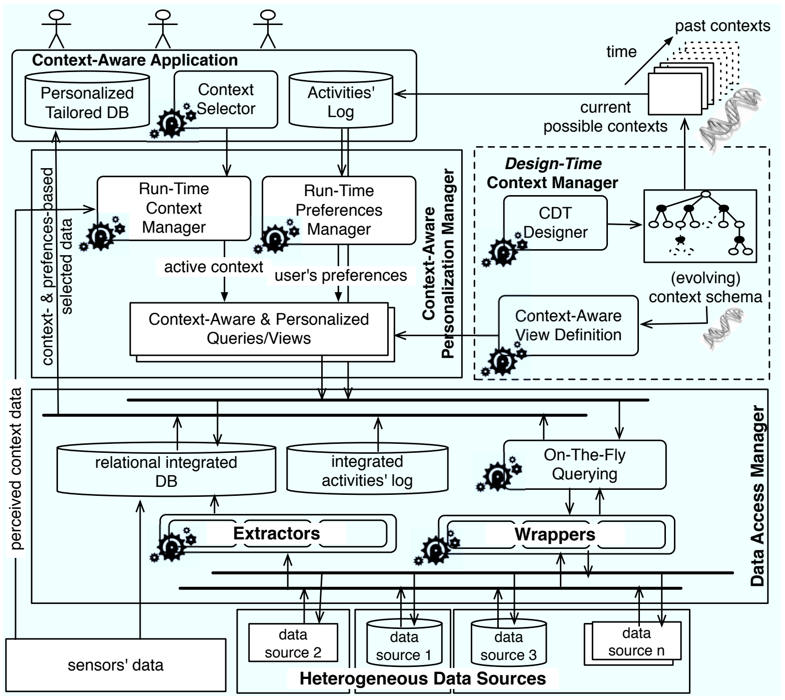
a) **Design-Time Context Manager**: in charge of supporting i) the design of the context schema of the specific application scenario by means of a suitable *context model*, and ii) for each context admitted by the context schema, its association with the appropriate *context-aware view* over the available data sources.

b) **Context-Aware Personalization Manager**: it determines, at each instant, the currently *active context*, applies contextualization to the queries issued towards the different data sources and delivers the context-aware data to the users and applications. Some of the aspects characterizing the current context can be automatically derived (such as the location and time of the user, her/his role within the application scenario, the kind of device used to access the data), while others may require an explicit specification by the user (for instance, her/his current topics of interests).

c) **Data Access Manager**: it offers access to data that can reside internally, for instance in a database, or in other, external data sources, providing an early-tailored, context-aware unified answer.

The components we have just described constitute the basic architecture of the Context-ADDICT project. While striving to design this architecture in detail and to adopt it in some real-life cases, we came across several of the research challenges that are currently absorbing our group. Let us consider the **Design-Time Context Manager**; here, the designer's role is to envisage the possible contexts the user will incur during the system's

[2]A context schema specifies the possible contexts for the current application domain.



Context-Aware Application
Personalized Tailored DB
Context Selector
Activities' Log
past contexts
time
current possible contexts
Run-Time Context Manager
Run-Time Preferences Manager
active context
user's preferences
Context-Aware & Personalized Queries/Views
Context-Aware Personalization Manager
context- & prefences-based selected data
Design-Time Context Manager
CDT Designer
(evolving) context schema
Context-Aware View Definition
perceived context data
relational integrated DB
integrated activities' log
On-The-Fly Querying
Extractors
Wrappers
Data Access Manager
data source 2
data source 1
data source 3
data source n
sensors' data
Heterogeneous Data Sources

Figure 1: A general view of the architecture of a data-oriented, context-aware system.

life. However the application requirements of a context-aware system are intrinsically dynamic and thus can evolve, therefore it is highly probable that, after some time, the context perspectives and parameters anticipated at design time be not applicable any more. Accordingly, the context schema used as the basis of the tailoring process should be allowed to change over time, along with the design and assignment of the related context-aware views. These two strictly related problems originated the research that is briefly described in Section 2. More about context-aware query tailoring will be described also in Section 4.

Note that the user interests and preferences may vary according to the context the user is currently in, for a change in context may change the relative importance of information. In the **Context-Aware Personalization Manager**, *contextual preferences* are used to refine the views associated with contexts, by imposing a ranking on the data of a context-aware view and adding the opportunity to dispose of the less interesting portion if needed. On the other hand, since we cannot actually expect a user to manually specify the long list of preferences that might be applied to all available data when a context becomes active, we propose a methodology and a system, PREMINE, where data mining is used to learn contextual preferences from the previous user's querying activity. This research is briefly described in Section 3.

The proliferation of freely-accessible data-intensive websites, as well as initiatives for open-accessible linked data in the Web [5], provide the users with potential sources of precious information. The **Data-Access Manager**, discussed in Section 4, supports the context-aware querying of heterogeneous and dynamic data sources, retrieving from the sources only the data that are consistent with the current context. Among the information sources we also consider devices which are spatially distributed and possibly mobile, the task of some of which is to monitor different kinds of physical phenomena for application support. Still at the data-access level, we designed the PerLa language and middleware which allow for declarative gathering of data from the environment, where some of these "pervasive" data are used to provide information to the operational system, while a context-management extension to the same language permits to declare the context and to provide the actual values for the context parameters.

2 Design-Time Context Manager

A key element in the design of context-aware systems is the representation and management of context and of its attributes, to be able to define a relationship between each context and the aspects of interest (such as relevant information, applicable "rules" and behaviour, presentation, etc.) of the application scenario. As all well-conceived formalisms, a model that allows the designer to specify such aspects should satisfy at least the requirements of generality (it should be suited to represent any application scenario), modularity (describing the context perspectives at the correct level of granularity), readability (to serve as design documentation), expressivity (allowing querying, reasoning or constraint specification on the contexts of a given target application) [6, 22].

We devoted our initial efforts to the definition of a suitable context model, the *Context Dimension Tree* formally presented in [7], used to specify at design-time the context schema, that is, all possible contexts the user will be acting in. The *context schema* (or *CDT*) of a given scenario is a hierarchical structure consisting of i) *context dimensions* (black nodes), modeling the different perspectives through which the user perceives the application domain and ii) the allowed *dimension values* (white nodes), i.e. the actualized parameters based on which the context-aware information, behaviours, etc. are to be selected. Each context is then defined as a combination of *context element*s, that is, <*dimension-value*> pairs. The adoption of a hierarchical organization allows us to employ different abstraction levels to specify and represent contexts. It is also possible to define appropriate constraints on the schema, that prevent some of the combinations of context elements, if meaningless for the current application. For example, consider the university campus scenario where users are students of any level (undergraduate, graduate, PhD), faculty members and guests. The university provides a mobile application to allow an easy, context-aware, personalized access to all information that might be of interest, like courses, schedules, rooms, other activities, local transportation, dining resources. Here, a valid *context* is the one of a student, who is interested in finding information about courses, formally expressed as $user = student \wedge interest\_topic = courses$, while the context $user = undergraduate \wedge interest\_topic = research$ might be considered meaningless by the designer and thus ruled out by writing an appropriate constraint.

It is easy to see that the burden of explicitly defining the portion of data deemed interesting for each context is a time-consuming task given the possibly high number of valid contexts; therefore the naive *per context* association of views to contexts evolved towards the definition of a methodology working on a *per context-element* basis, which supports the designer in the definition of a view for each context element – called *relevant area*, or $\mathcal{R}el$ –; based on this specification, the system automatically derives the data portion associated with a context by opportunely combining the relevant areas of the involved context elements. The advantage is twofold: i) to reduce the designer's work and ii) to achieve a high flexibility w.r.t. an evolving application scenario. The approach is fully general, and has been profitably applied to other data models; in particular, in order to add the necessary reasoning capabilities, we have explored the representation of the context model by means of ontologies [23] and logic programs [24]. For data tailoring in the relational model [8], we have defined operators that apply to sets of relations and support the composition of context-aware views. As a result, the designer's effort is devoted to the definition of the CDT and of the $\mathcal{R}el$s, while a tool[3] performs, for each context, the automatic computation of the resulting context-aware views.

Starting from techniques devised in the literature on schema evolution in various fields [2, 19], our recent research proposes strategies to flexibly manage the evolution of the context schema. During system operation, the initial vision of the designer about the possible contexts pertaining to a given application scenario may change, and new dimensions or values might come into play or old ones might become not applicable. As a result, the context schema has to change, and for each new node the associated view needs to be computed or a modification of some existing views might be necessary. For instance, in the university scenario, a change from quarters to semesters (i.e., a change of dimension values) would cause a revision of the schema and the need to

[3]The activities of the Design-Time Context Manager are supported by *CADDFrame*, integrating a CDT Designer to define the CDT, a tool to specify the $\mathcal{R}el$s, and an engine that combines them to compute the final views associated with each valid context.

write new views, associated with the semester data. A context schema change might also be triggered when the user device detects some environmental parameter values that are not envisaged by the current schema.

The Design-Time Context Designer supports context schema modifications by allowing the use of a predefined set of *evolution operators* that keep the coherence between the schema changes and the (induced) changes in the encompassed contexts. The sequence of the modifications is logged, and used to refresh the user context and views in such a way that, when the user device detects a context change and requests the related data, these are delivered consistently with the new schema. This is made possible by the fact that the per-element view-construction strategy is employed, and thus only a limited number of views – those related to the nodes that are involved in the evolution operation – have to be added or revised by the designer.

3 Context-Aware Personalization Manager

Data personalization based only on the notion of context represents but a partial solution to the problem of information reduction, since the tailoring turns out to be often too coarse-grained. For example, consider the context of Bob, a student interested in finding a free room for the afternoon; a contextual system will suggest only the rooms close to his current location, and will not be able to propose any ranking or further filtering of the contextual data according to Bob's tastes. Actually, the context often plays a crucial role in determining different prefences of the same person: for instance Bob could be very interested in quiet rooms when he is alone, and in open spaces when he is with his friends.

To obtain a more effective personalization, the *Run-Time Preference Manager* (see Figure 1) couples the knowledge of context with the user personal preferences: this allows to specify which information represents the actual interests and needs for Bob, when he is in a particular context (alone or with friends). In this direction, some approaches [29, 21, 17, 20] have been proposed for personalizing relational data (both tuples and attributes) on the basis of contextual preferences defined by the user. However, when considering a large variety of data and a rich set of possible contexts, the manual specification of an extensive list of preferences may be onerous for the user, who may be discouraged w.r.t. the activity of explicitly indicating his/her preferences. A way around this problem is to learn the user tastes from a log of his/her past behaviour, i.e., the past data-querying activities. We use data-mining algorithms to discover implicit contextual behaviours, in the form of *association rules* that correlate each contexts with the values accessed in it. In particular, a σ-rule $\bar{r}_\sigma$ on the relation schema $R(X)$ is a triple $\langle C \rightarrow cond, sup, conf\rangle$, where $C \rightarrow cond$ is an association rule, C is a context and $cond$ a conjunction of simple conditions in the form $A = value$, with $A \in X_i$ for some $R_i(X_i) \in R(X)$; sup and $conf$ indicate the support and the confidence of $C \rightarrow cond$, respectively. The support corresponds to the frequency of $C \wedge cond$ be true in the log storing the user activities; the confidence corresponds to the (conditional) probability of finding $cond$ true in the log of the activities done in context C, and is given by $\frac{sup(C \wedge cond)}{sup(cond)}$. For example, if we consider the table CLASSROOM(NAME,BUILDING,FLOOR,LOCATION,TYPE) in the dataset of our scenario, and we mine the σ-rule $\langle user = student \wedge situation = alone \rightarrow \sigma_{type=computerized} CLASSROOM, 0.8, 0.9\rangle$ by analysing Bob's activities, we can state that, when alone, he is often interested in computer-equipped classrooms (with a support 0.8 and a confidence 0.9).

σ-rules are used to learn preferences on tuples, but the preferences can be mined also on attributes. A π-rule $\bar{r}_\sigma$ on the relation schema $R(X)$ is a triple $\langle C \rightarrow A, sup, conf\rangle$, where C is a context and $A \in X_i$ for some $R_i(X_i) \in R(X)$. For example, the π-rule $\langle user = student \wedge situation = alone \rightarrow \{location\}, 0.7, 0.8\rangle$ states that Bob, when alone, often visualizes the classroom location.

Our approach, differently from the majority of recommendation systems, does not require any explicit input from the user about his/her preferences; we are studying its seamless combination with recommendations that can be used to personalize the views for new users, when the history of their activities is not available yet, on the basis of the behaviours of other, "similar" users in the same context.

4 Data-Access Manager

In Context-ADDICT the data sources are generally *dynamic*, *transient*, and *heterogeneous* in both their data models (e.g., relational, XML, RDF) and schemas[4]. Therefore, in order to access their content, it is first necessary to understand their schemas and to represent them in a suitable semantic formalism (e.g., description logics or Datalog-like languages) to enable their semi-automatic reconciliation. We designed *extractors* (Figure 1) to automatically derive, through reverse-engineering, the *external schemas* from the schemas of the data sources. In particular the extractors are *domain-aware*, i.e., they include in the external schema only the data-source entities which have a counterpart in a global schema [12] which is used as a uniform representation of the available information in the integration system. Each external schema is then (semi-)automatically mapped to the global schema, which can be eventually queried using a suitable query language, e.g. SQL or SPARQL, depending on the model chosen for the global schema. The mappings between the external schemas and the global schema are provided in a semi-automatic way, using schema matching tools [3].

In Context-ADDICT, the schemas are expressed using the CA-$\mathcal{DL}$ data language that allows to: (i) uniformly represent the data and the user context(s) in any application domain and (ii) support efficient context-aware query distribution and answering, for external data-sources context-based information filtering is performed at run-time (*early tailoring*), by selecting fragments of the global schema that are considered relevant for the current context. These relevant areas are computed by means of the per-context-element strategy discussed in Section 2. Each relevant area is matched against the external schemas produced by the extractors, inducing, through the mappings, corresponding context-aware fragments of the external schemas [23]. Data access proceeds as follows: the queries formulated over the global schema are first translated into context-aware queries using the specifications provided by the relevant areas. Then, the *contextualized queries* are reformulated in terms of the global schema and of the mappings, producing queries over the data sources. The queries are then distributed to the corresponding data sources and finally translated into the native language of each target source by means of suitable wrappers.

A similar perspective is adopted to access data coming from sensors. We developed PerLa[5](PERvasive LAnguage) [28] a declarative, high-level language that allows to query a pervasive system within the unifying setting provided by Context-ADDICT, hiding the difficulties related to the need of handling different technologies. A database-like abstraction of the whole sensor network is provided in order to hide the high complexity of low-level programming, allowing users to retrieve both functional and non-functional data from the system and send configuration/activation commands to the sensors in a fast and easy way. A middleware provides an abstraction for each device in terms of a *logical object*, and supports the execution of PerLa queries. During the design of the middleware, we strove to make the definition and the addition of new devices easier by minimising the amount of low-level code the user has to write to make the new device recognizable by the system.

Context awareness has a two-fold role with respect to sensors' data: part of the collected information contributes to the information base to be made available through the application and is filtered according to context as in the general case; part of it must be directly used to determine the context itself, such as the time of the day or the location (e.g., from GPSs). In the latter situation, the information is directly exploited by the *Run-Time Context Manager* (see Figure 1) to actualise the correct context. PerLa, originally conceived only for querying sensors, has thus been extended with statements to i) *define the CDT structure*, also with the capability of acquiring that part of context information that cannot be deduced from sensor readings; ii) *create a context* on a defined CDT; iii) *activate/deactivate a context* at run-time as by the actual values of the context variables; iv) *perform the contextual actions* required on the system, e.g.: activating actuators, changing measurement modalities, cutting and tailoring queries from a general query stereotype.

[4]In our research we mostly concentrate on data sources whose schema is available, while for less structured data sources we resort to techniques found in state-of-the-art literature [15].

[5]http://perlawsn.sourceforge.net/index.php

5 Conclusions

We have given a brief report of our understanding of the research on context-awareness, and of the activities on context-aware information systems going on within the PEDiGREE group at Politecnico di Milano. The overall research area is so stimulating that we continuously encounter topics that should be investigated, like context sharing, automatic synthesis (from observing the environment) of the possible contexts and of the views to be associated with them, automatic recognition of the next active context to the end of anticipating critical situations [13], uncertainty and inconsistency of the context data, stability of the system when the context changes become too frequent for adaptation, support of non-functional requirements like context-aware data quality [4], and many others, which can be a stimulus to the scientists who are attracted by this ever-growing research area.

References

[1] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Intl J. of Ad Hoc Ubiquitous Computing*, 2(4):263–277, 2007.

[2] A. Balmin, Y. Papakonstantinou, and V. Vianu. Incremental validation of XML documents. *ACM Trans. on Database Systems*, 29(4):710–751, 2004.

[3] Z. Bellahsene, A. Bonifati, and E. Rahm. *Schema Matching and Mapping*. Springer-Verlag, Heidelberg (DE), 2011.

[4] L. Bertossi, F. Rizzolo, and L. Jiang. Data quality is context dependent. To appear in *Proc. of the 4th Intl Work. on Business Intelligence for the Real Time Enterprise*, 2010.

[5] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[6] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A data-oriented survey of context models. *SIGMOD Record*, 36(4):19–26, 2007.

[7] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. Context information for knowledge reshaping. *Intl J. of Web Engineering and Technology*, 5(1):88–103, 2009.

[8] C. Bolchini, E. Quintarelli, and R. Rossato. Relational data tailoring through view composition. In *Proc. 26th Intl Conf. on Conceptual Modeling, ER*, pages 149–164, 2007.

[9] P. Brézillon and S. Abu-Hakima. Using knowledge in its context. *AI Magazine*, 16(1):87–91, 1995.

[10] S. Buchholz, T. Hamann, and G. Hübsch. Comprehensive structured context profiles (CSCP): Design and experiences. In *Proc. of 1st Intl Work. on Context Modelling and Reasoning*, pages 43–47, 2004.

[11] H. Chen, T. Finin, and A. Joshi. An intelligent broker for context-aware systems. In *Proc. of Intl Conf on Ubiquitous Computing - Poster Session*, pages 183–184, 2003.

[12] C. A. Curino, G. Orsi, E. Panigati, and L. Tanca. Accessing and documenting relational databases through OWL ontologies. In *Proc. of 8th Intl Conf. on Flexible Query Answering Systems*, pages 431–442, 2009.

[13] Y. Ding, H.R. Schmidtke, and M. Beigl. Beyond context-awareness: context prediction in an industrial application. In *Proc. of the 12th Intl Conf. on Ubiquitous Computing*, pages 401–402, 2010.

[14] Barbara Pernici (ed.). *Mobile Information Systems - Infrastructure and Design for Adaptivity and Flexibility*. Springer-Verlag, Heidelberg (DE), 2006.

[15] G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca. The Lixto data extraction project - back and forth between theory and practice. In *Proc. of the 23rd Symp. on Principles of Database Systems*, pages 1–12, 2004.

[16] T. Gu, H. K. Pung, and D. Q. Zhang. A service-oriented middleware for building context-aware services. *J. of Network and Computer Applications*, 28(1):1–18, 2005.

[17] E. Jembere, M. O. Adigun, and S. S. Xulu. Mining context-based user preferences for m-services applications. *Web Intelligence*, pages 757–763, 2007.

[18] M. Kaenampornpan and E. O'Neill. An integrated context model: Bringing activity to context. In *Proc. of Work. on Advanced Context Modelling, Reasoning and Management*, 2004.

[19] M. M. Lehman. Software's future: Managing evolution. *IEEE Software*, 15(1):40–44, 1998.

[20] J. J. Levandoski, M. F. Mokbel, and M. E. Khalefa. Caredb: A context and preference-aware location-based database system. *PVLDB*, 3(2):1529–1532, 2010.

[21] A. Miele, E. Quintarelli, and L. Tanca. A methodology for preference-based personalization of contextual data. In *Proc. of the 12th Intl Conf. on Extending Database Technology*, pages 287–298, 2009.

[22] A. Mileo, D. Merico, and R. Bisiani. Support for context-aware monitoring in home healthcare. In *Intelligent Environments (Workshops)*, pages 177–184, 2009.

[23] G. Orsi. *Context Based Querying of Dynamic and Heterogeneous Information Sources*. PhD thesis, Politecnico di Milano, 2011.

[24] G. Orsi and L. Tanca. Context modelling and context-aware querying: can datalog be of help? To appear in *Proc. of the Datalog 2.0 Workshop*, 2010.

[25] D. Preuveneers, J. van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, E. Berbers, K. Coninx, and K. de Bosschere. Towards an extensible context ontology for ambient intelligence. In *Proc. of the 2nd European Symp. on Ambient Intelligence*, pages 148–159, 2004.

[26] P.G. Raverdy, O. Riva, A. de La Chapelle, R. Chibout, and V. Issarny. Efficient context-aware service discovery in multi-protocol pervasive environments. In *Proc. of 7th Intl Conf. on Mobile Data Management*, pages 3–11, 2006.

[27] Y. Roussos, Y. Stavrakas, and V. Pavlaki. Towards a context-aware relational model. In *Proc. of 1st Intl Context Representation and Reasoning Work.*, pages 7.1–7.12, 2005.

[28] F. A. Schreiber, R. Camplani, M. Fortunato, M. Marelli, and G. Rota. Perla: A language and middleware architecture for data management and integration in pervasive information systems. To appear in *IEEE Trans. on Software Engineering*, 2011.

[29] K. Stefanidis, E. Pitoura, and P. Vassiliadis. Adding context to preferences. In *Proc. Intl Conf. Data Engineering*, pages 846–855, 2007.

[30] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Intl Work. on Advanced Context Modelling, Reasoning and Management*, pages 31–41, 2004.

[31] M. Theodorakis, A. Analyti, P. Constantopoulos, and N. Spyratos. A theory of contexts in information bases. *Information Systems*, 27(3):151–191, 2002.