An Evaluation of Query Languages for Context-Aware Computing

Pari Delir Haghighi, Arkady Zaslavsky, and Shonali Krishnaswamy

Center for Distributed Systems and Software Engineering

Monash University, Australia

{Pari.DelirHaghighi, Arkady.Zaslavsky, Shonali.Krishnaswamy}@infotech.monash.edu.au

Abstract

Context-awareness highly depends on easy and prompt access to the right information and this needs overcoming challenges of query expression and processing of context information. One of the main considerations in querying context is selecting and applying an appropriate and expressive query language for defining user's context needs. This paper presents an overview of current languages used for querying context and investigates their strengths and limitations in expressing different aspects of context.

1. Introduction

Generic infrastructures for context-awareness facilitate context management by collecting context information from different context sources and making it available to context-aware applications. This type of infrastructure is also referred to as a context service because the main task of a context service is to serve clients' context needs that are expressed as context queries [1]. However, to achieve this goal, context services need to overcome the challenges of querying context in pervasive computing environments. One of these challenges is selecting and applying an appropriate and expressive query language for defining user's context needs.

This paper is structured as follows: Section 2 reviews the main characteristics of context. Section 3 presents a classification of languages used for querying context in current context-aware systems. Section 4 compares these languages with respect to the main aspects of context and reports on the results of this evaluation. Finally section 5 concludes the paper.

2. Different Aspects of Context

The most adopted definition of context has been presented by Dey et al. [2] that defines context as "any information that can be used to characterize the situation of an entity". This information can be sensed, derived, measured, detected, reasoned, computed, calculated or explicitly entered by users. Different aspects of context can be summarized as follows:

Dynamic vs. static

- Context can be dynamic like location or it can be static like user name;

Continuous Data Streams

- Context can be continuous data streams (e.g. sensor readings);

Metadata and Quality of Context (QoC)

- Context can be associated with metadata;
- Context can be temporal [3];
- Context is liable to imperfection and can be erroneous, ambiguous, unavailable or incomplete [3, 4];
- The same context information about an entity can be different in QoC such as precision [5];

Proximity and spatial relationships

- Context can be spatial [3];
- Context can express proximity;
- Context information can have dimensions such as longitude, latitude and altitude;

Situations and emotions

- Context information can be a situation that is derived and reasoned from other context information (e.g. 'John is busy');
- Context can be difficult to capture such as user emotions;

Context-aware applications need to define different characteristics of context in a query to access certain context information they require. Next section provides a classification of current languages used for querying context and describes the systems and infrastructures that apply them.

3. Categorization of Languages for Querying Context

Query languages are pivotal for querying context and determine the way queries are expressed and what information needs to be obtained.

There are a variety of query languages that have been employed in context-aware applications and infrastructures for querying context information. Some implementations have used existing query languages to

access information or extended them such that they are tailored to context query needs while others have introduced new languages. Considering their common characteristics, languages for querying context can be grouped into five main categories of SQL-based, Graphical, XML-based, RDF-based and programming languages.

3.1. SQL-based Query Languages

One of the most common and powerful languages used for making queries over relational databases is SQL. SQL is a well-established, declarative query language that provides a solid basis for creating and executing wide range of queries and is supported as a standard by many existing implementations.

One of the SQL-based works is CIS (Contextual Information System) [6] that provides applications with easy access to context information. CIS presents providers of static data with a Contextual Service Interface-SQL (CSInt-SQL) wrapper using a database without any need for coding and it offers providers of dynamic context with a database-like interface [6]. The primary function in the interface is called Query function that its arguments allow user to express important aspects of context such as temporal context and metadata including quality of context.

Another SQL-based approach is the context management system [7] for CML (Context Modeling Language). This work enables mapping the CML models to relational data and into a format that can be queried by SQL. A text-based context schema has been developed to capture CML models and a context schema compiler is used to validate and process the context model. The context schema can automatically be mapped into a SQL database via a set of tools [7].

The CML model is an expressive and powerful model that targets most aspects of context including imperfect context but the context management system does not provide any solutions for addressing unavailable or imperfect context.

Sensors are one of the important sources of context that produce data in streams. One of the SQL-based works for sensor queries is the Fjords architecture [8] that manages multiple queries over streaming sensor data. A Fjord (Framework in Java for Operators on Remote Data Streams) is a query plan data-structure that enables users to create queries by combining push-based sensor data with traditional pull-based data [8].

The main focus of the Fjords architecture is querying sensor data streams without addressing other types and aspects of context queries but this work demonstrates that although SQL is originally for persistent data, there are possibilities of using and adjusting it to query non-persistent data.

3.2. Graphical Query Languages

There are other alternative query languages to SQL that can be used for querying relational models. One of these languages is QBE (Query-By-Example) that is based on domain relational calculus. QBE is a user-friendly and graphical relational language that enables users to make queries by creating example tables.

Spatial mQBE [9] is a language for Relational Database Management System (RDBMS) that extends QBE by supporting both spatial and non-spatial queries on mobile devices. The Spatial mQBE attempts to provide a QBE user-interface that makes entering queries easier than writing a textual language as SQL. However, the interface requires user to have a basic knowledge of algebra and set operations to enter the spatial conditions. Furthermore, the interface limits the user entries to the attributes and their values (through the pull-down menu) on the screen.

Another work in querying context that aims to provide a user-interface for querying a relational schema is Chiromancer [10] that uses the Query by Browsing (QBB) paradigm and the materialized views concept. Chiromancer is a visual query language (VQL) that enables expressing of context queries on handheld devices using familiar concepts from the desktop user-interface paradigm. Chiromancer represents both schema and queries as a folder hierarchy and each folder as a materialized view on the stored data.

Compared to the QBE's interface, the Chiromancer provides user with a more flexible interface that offers extra functionalities, navigation and selection choices.

3.3. XML-based Query Languages

One of the important works that applies XML-based language for querying context is the Nexus architecture [11, 12]. Nexus is an open platform that facilitates developing location-aware applications and enables integration of and interaction between the applications. The Nexus platform is based on a common augmented world model that is described by AWML (Augmented World Modeling Language) and can be queried using AWQL (Augmented World Querying Language). The augmented world model represents the world as data objects with attributes and all the objects produced by a context provider belong to an Augmented Area. Context providers register their Augmented Areas and their object types with the Area Service Register that will assist the system with queries. An extension to AWML has integrated metadata into the model to facilitate resource finding, context selection, context quality and data processing [12].

Some of the strengths of AWQL queries are their support for generalization and aggregation rules, nearest

neighbor queries and spatial relationships [11-13]. Other advantage of AWQL queries is that they can be mapped into SQL queries using multiple joins. Despite expressiveness of XML, this language does not provide sufficient flexibility to support complex queries and expression of different aspects of context.

Another XML-based language is the graph-oriented CoCo (Context-Composition) language [14] developed for the CoCo infrastructure. In the CoCo language, a context information object (CI-object) describes an entity at a point in time in a specific scale (format) and the aspects of entities are represented using a context information class (CI-class). Since CoCo documents support the replaceable-by and approximated-by dependencies, if the required information delivery from a source fails, the CoCo controller that executes CoCo documents allows dynamic binding and automatic adaptation to access the requested information from other available context sources at runtime [14].

3.4. RDF-based Query Languages

The SOCAM (Service-oriented Context-Aware Middleware) architecture [15, 16] describes context queries in OWL that is a RDF-based language and supports complex context queries by combining predicates and Boolean algebra. Although the structure of RDF-based languages is more suitable for context queries, it is not as flexible as SQL and it can result in large amounts of overheads as it requires creation of many classes, subclasses and relationships [17].

One of the innovative works in the area of querying context is CQP (Collaborative Query Processing) protocol that has been implemented on top of MoGATU framework [17]. MoGATU is a profile-driven and proactive peer-to-peer semantic data management system for pervasive environments that defines queries in DAML-OIL [18]. The CQP model is based on the concept of nested-loop joins and the principles of Contract Nets [17]. Using the CQP protocol, mobile devices can query their neighboring devices by decomposing queries and assigning them to different peer devices. The CPQ model supports both explicit queries asked by users and implicit queries that are inferred from user profiles.

The CQP seems to be a promising work that enables to overcome the obstacles of context acquisition and queries in ubiquitous computing environments. However, it requires mobile devices to have some knowledge of ontologies and to be able to parse DAML+OIL information. The context acquisition in CQP is mostly based on peer mobile devices and this creates the question of availability and caching capabilities of these devices.

3.5. Programming Query Languages

One of the works in the area of querying continuous context information is the iQL [19] programming model and language that addresses the special characteristics of networked data sources. The iQL language aims to facilitate management of context sources and allows runtime rebinding to new sources if old sources fail or degrade.

The iQL language consists of entities called composers. Composers have a value that is computed from input values and is determined by a composer specification. The iQL 'input' expression allows rebinding to new advertised context sources that satisfies the query and describes what information needs to be retrieved rather than how it is done.

Classification of languages for querying context allows a comparison of these languages and provides an insight into strengths and limitations of each language. Next section presents an evaluation of aforementioned query languages.

4. A Comparative Evaluation of Languages for Querying Context

This section reports on the results of a comparative evaluation of languages used for querying context in current context-aware systems and infrastructures. The motivation behind this evaluation is to compare query languages used for querying context and discover if these languages are able to express different characteristics of context. Table 1 summarizes the results of this evaluation and illustrates the strengths and limitations of each language with respect to the main aspects of context. The summary of results is as follows:

- **SQL-based languages:** these languages are able to support all the main characteristics of context in the criteria. Originally, SQL has not been developed for querying context in mobile and pervasive environments but the extensions that have been added to SQL have addressed these limitations.
- **Graphical query languages:** this category has fulfilled fewer criteria than the rest of the languages. In graphical approaches, the size of the displays in mobile devices makes it difficult to express various context needs and complex queries. The problem of expressing queries can be avoided to some point by displaying symbols and small icons like the icons used in the Chiromancer [10] interface for each category of context.

Table 1. Evaluation of languages for querying context

Languages for querying context	Systems	Aspects of Context					
		Dynamic vs. Static	Continuous data streams	Metadata (i.e. QoC, temporal)		Spatial	Situations
SQL-based	CIS	good	none	good	good	average	none
	CMS	good	none	good	good	average	good
	Fjords	average	good	none	good	none	none
Graphical	Spatial mQBE	none	none	none	none	good	none
	Chiromancer (QBB)	good	none	average	good	average	none
XML-based	Nexus (AWQL)	poor	none	good	good	good	none
	CoCo	poor	none	good	good	poor	none
RDF-based	SOCAM (OWL)	average	none	average	good	average	good
	MoGATU-CQP (DAML-OIL)	good	none	average	good	average	good
Programming	iQL	poor	good	poor	good	average	none

- **XML-based languages:** while not being as developed as SQL language, XML-based languages have realized many requisite criteria. However, at this stage of their development, these languages lack the support for continuous data streams and situations. XML is regarded more as a representation language for declaring and expressing simple data structures than a language for representing complex knowledge [20].
- **RDF-based languages:** ontology based languages applied in SOCAM and MoGATU have fulfilled most of the requirements but they do not provide support for continuous data streams. One of the strengths of the RDF-based languages is their ability to describe context semantically and define context at a high level; therefore, different aspects of context such as situations can be expressed easily and in simpler fashion in OWL or DAML-OIL than in other languages.
- **Programming query languages:** the iQL language has also met most of the criteria except for the situations and quality of context. Programming languages have the capabilities of addressing various computable queries but they require developing complex algorithms to process the queries.

The results of our evaluation showed that the SQL and RDF-based languages are the most well-developed languages in terms of expressive power for specifying different needs and aspects of context. This obviously includes the caveat that the specific individual features supported are largely dependent on systems. In terms of our evaluation, the CML system (SQL-based) and the MoGATU system (RDF-based) are the most sophisticated systems that have addressed most of the main aspects of context.

5. Conclusion

Querying context encompasses many issues and complexities. One of these issues is applying a query language that is able to express different characteristics of context. This paper provided a classification of languages used in current systems for querying contextual information and evaluated these languages with respect to the main aspects of context.

There are many open issues that need to be addressed in terms of developing both languages and systems for querying context. These include: mappings queries from one language to another thereby facilitating interoperability between queries expressed in different languages, flexible and natural languages for querying contextual information, systems that support *adaptive*, *proactive* and *intelligent query processing* and further development of current query languages with a focus on simplicity and expressive ease.

6. Bibliography

- [1] M. Ebling, Hunt, G. and Lei, H., "Issues for Context Services for Pervasive Computing," In Proceedings of Middleware'01, Advanced Workshop on Middleware for Mobile Computing, Heidelberg, Germany, November 2001.
- [2] A. Dey, and Abowd, G., "Towards a Better Understanding of Context and Context-Awareness," In the Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, April 2000.
- [3] A.H. van Bunningen, Feng, L. and Apers, P.M.G., "Context for Ubiquitous Data Management," In the Proceedings of the 2005 International Workshop on Ubiquitous Data Management (UDM'05), Tokyo, Japan, April 2005.
- [4] K. Henriksen, and Indulska, J., "Modelling and Using Imperfect Context Information," In Proceedings of the 2nd IEEE

Annual Conference on Pervasive Computing and Communications. Workshop on Context Modelling and Reasoning (CoMoRea'04), Orlando, Florida, March 2004.

[5] T. Buchholz, Kupper, A. K. and Schiffers, M., "Quality of Context: What It Is and Why We Need It," In Proceedings of the Workshop of the HP OpenView University Association 2003 (HPOVUA 2003), Geneva, Switzerland, July 2003.

[6] G. Judd, and Steenkiste, P., "Providing Contextual Information to Pervasive Computing Applications," In Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), The University of Texas, Arlington, March 2003.

[7] T. McFadden, Henricksen, K. and Indulska, J., "Automating Contextaware Application Development," First International Workshop on Advanced Context Modelling, Reasoning and Management (UbiComp'2004), Nottingham, September 2004.

[8] S. Madden, and Franklin, M. J., "Fjording the stream: An Architecture for Queries Over Streaming Sensor Data," In Proceedings of the 18th International on Data Engineering, San Jose, California, February 2002.

[9] S.A. Rahman, and Bhalla, S., "Supporting Spatial Data Queries for Mobile Services," In Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), University of Technology of Compiègne, France, September 2005.

[10] S. Polyviou, Evripidou, P. and Samaras, G., "Context-Aware Queries Using Query by Browsing and Chiromancer," Second International Conference on Pervasive Computing, Vienna, Austria, April 2004.

[11] M. Bauer, Becker, Ch. and Rothermel, K., "Location Models from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks," Personal and Ubiquitous Computing, vol. 6, pp. 322-328, 2002.

[12] N. Hönle, Käppeler, W., Nicklas, D. Schwarz, Th. and Grossmann, M., "Benefits of Integrating Meta Data into a Context Model," In Proceedings of the 3rd International Conference on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops), Kauai Island, HI, March 2005.

[13] M. Grossmann, Bauer, M. and Honle, N., "Efficiently Managing Context Information for Large-scale Scenarios," In Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), Kauai Island, HI, March 2005.

[14] T. Buchholz, Krause, M., Linnhoff-Popien, C. and Schiffers, M., "CoCo: dynamic composition of context information," The 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2004), Boston, Massachusetts, August 2004.

[15] T. Gu, Pung, H.K. and Zhang, D.Q., "A Middleware for Building Context-Aware Mobile Service," IEEE 59th Vehicular Technology Conference, Milan, Italy, May 2004.

[16] T. Gu, Pung, H.K. and Zhang, D.Q., "Toward an OSGi-based infrastructure for context-aware applications," *Pervasive Computing*, vol. 3, pp. 66 – 74, October-December 2004.

[17] F. Perich, Joshi, A., Yesha, Y. and Finin, T., "Collaborative Joins In a Pervasive Computing Environment," The International Journal of Very Large Databases, vol. 14, pp. 182-196, April 2005.

[18] F. Perich, Joshi and A., Yesha, Y., "On Data Management in Pervasive Computing Environments," IEEE Transactions on Knowledge and Data Engineering, vol. 16, pp. 621-634, May 2004.

[19] N. H. Cohen, Lei, H., Castro, P., Davis II, J.S. and Purakayastha, A., "Composing Pervasive Data Using iQL," In Proceedings of 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002), Callicoon, New York, June 2002.

[20] H. Chen, Perich, F., Finnin, T. and Joshi, A., "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications," In Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), Massachusetts, August 2004.