View metadata, citation and similar papers at core.ac.uk brought to you by CORE
provided by Repositório Institucional da Universidade de Aveiro

# Scalable Semantic Aware Context Storage

Mário Antunes
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal
Email: mario.antunes@av.it.pt

Diogo Gomes
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal
Email: dgomes@av.it.pt

Rui Aguiar
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal
Email: ruilaa@av.it.pt

***Abstract*—In recent years the Internet has grown by incorporating billions of small devices, collecting real-world information and distributing it though various systems. As the number of such devices grows, it becomes increasingly difficult to manage all these new information sources. Several context representation schemes have tried to standardize this information, however none of them have been widely adopted. Instead of proposing yet another context representation scheme, we discuss an efficient way to deal with this diversity of representation schemes. We define the basic requirements for context storage systems, analyse context organizations models and propose a new context storage solution. Our solution implements an organizational model that improves scalability, semantic extraction and minimizes semantic ambiguity.**

***Keywords*—Internet of things, M2M, context information**

## I. Introduction

When we think about the Internet we mostly consider servers, laptops, routers and fixed broadband that have almost reached every household. But the fact is the Internet is growing very fast as we speak in to new kinds of devices. Everyday devices (from mobile devices to environmental sensors networks) are now connected to the Internet, sharing massive amounts of information. According to the ICT Knowledge Transfer Network (ICT KTN), the number of mobile devices is expected to increase worldwide from 4.5 billion in 2011 to 50 billion by 2020 [1].

As microcosms of the Internet of Everything (IoE), cities stand to benefit the most from the untapped information shared by all these devices. Smart cities means many things to many people. Yet, one thing remains constant: part of being "smart" is utilizing information and communications technology and the Internet to address urban challenges. In smart cities, an entity's context can be used to improve efficiency, optimize resources and detect anomalies. The following example illustrate the importance of context information for smart cities. Before leaving home a citizen can reserve a parking space close to their destination. While they drive, the vehicle dynamically selects the best course. The selection not only takes into account traffic congestion but also road conditions, road accidents and other anomalies. A vehicle, upon approaching a semaphore, can announce its presence. With this information the semaphore can minimize the delay of each vehicle. Ambulances and other emergency vehicles have priority over the remaining, as such the semaphore automatically grants them a green light. At home the fridge connects to several supermarkets and produces a shopping list based on its contends and in the prices announced for each product.

In summary context information has the potential to dramatically improve the efficiency of smart cities. All the citizens benefit from the shared information without the need to manually gather it. Context information can be gathered, processed and shared among citizens, services and other devices.

For scenarios like the previous one to become reality, it is necessary to develop a way to manage such diverse machine made context information. One of the challenges is to store massive amounts of context information and provide a discriminative retrieval process.

Without loss of generality let us only consider pothole detection (a task of utmost importance for city officials). In an ideal context storage system, information related with road conditions should be automatically tagged with an appropriate tag. The information published by various sensors (on board of multiple vehicles, of multiple brands) does not explicit mention road condition's, it only contains measurements related with the vehicle. It is therefore necessary to allow search with concepts instead of simple words. It is quite difficult to add these functionalities to standard full-text search engines (present in several databases).

This is the main focus of this paper. The common definitions of context information [2], [3] does not provide any insight about its structure. In fact, each device can share context information with a different structure. E.g. sensory information and location information can be used to characterize an entity context, yet the two can have very different structures.

The main objective of context representation is to standardize the process of sharing context information through several services. Context-aware platforms strongly benefit from an uniform environment: the storage process is easier (the information follows a known structure) and the analysis of the information becomes simpler. Standard context management platforms commonly store context information in relational databases. We can devise a mapping process easily only if there is a common context representation. Multiple context representations have been proposed, such as ContextML [4], SensorML [5] COBRA-Ont [6], OASIS XDI [7] and OASIS XLIFF [8]. All these representations try to solve the same problem, but each representation is quite different and incompatible with the other. None of the above mentioned representations have been widely accepted either by the academia or the industry. Usually, each context-aware platform defines its own context representation based on the platform scenarios. This breaks compatibility between platforms and limits the quantity of context information that can be used in M2M applications.

It is possible (but unlikely), that in the future a context representation standard will be widely adopted. Until then, context-aware platforms have to deal with multiple context representations. The work presented in this paper addresses this problem and analyses possible representation schemes independent of storage solutions.

The remainder of the paper is organized as follows. In Section II we analyse how context information can be organized and define the basic requirements for context storage solutions. Two context organization models are proposed and analysed in Section III. We studied the organization models' impact on context information solutions in Section IV. Section V contains implementation details of our context storage solution. The results of the organization models evaluation are in Section VI. Finally the discussion and conclusions are presented in Section VII.

## II. Context organization

Context information is an enabler for deeper and further data analysis, requiring the integration of an increasing number of information sources. As previously mentioned, nowadays no widely accepted context representation scheme exists; instead there are several approaches to deal with context information.

We identified three different approaches. Previous works have adopted existing context representations [4], [9], [10]. These representations were optimized for that specific scenario, however this approach limits the quantity of context sources. Later on the authors recognized that the usage of a single context representation limits the context expressiveness [10].

Another possibility would be employing ontologies to normalize the storage process. For each context representation scheme there is an ontology that maps the representation into the internal data model [11]. This type of platform supports several context representations, yet it is necessary to define a new ontology for each representation.

Finally, we can accept the diversity of context representation as a consequence of economic pressures, and prepare for this inevitability.

According to the authors [12]–[14], the best solution to classify context information is through bottom-up characterization. Bottom-up characterization is massively dimensional, and there is no global consistency imposed by current practice. Although sensor information is not tagged by users, we can model the tagging process as keyword extraction [15]–[17]. A keyword is a sequence of one or more words that provide a compact representation of a document's content. Ideally, keywords represent in condensed form the essential content of a document.

A context storage solution must fulfil 3 requirements ([18]): ability to scale, generalize storing process and discriminative retrieval. Since the number of sensors is rapidly increasing, the quantity of context information is also increasing, and a context database must be robust to this increase. Simple replications allows a database to scale retrieve operations. A database system can be distributed through several nodes in order to improve write operations. Each node contains a set of the whole database (horizontal partitioning/sharding [19]). The last two requirements complement each other. In other words, the ideal context database must store and accurately pinpoint any piece of context information associated with any type of sensor. The most common methods to implement discriminative retrieval are through semantic web or information retrieval (IR) system. Since semantic web methods require ontologies (which in this scenario is a disadvantage) a context database must provide a discriminative retrieval based on IR systems.

## III. Context organization model

The common definitions of context information [2], [3] are so broad that any information related to an entity can be considered context information. These definitions also do not provide any insight about the structure of context information. From now on we will refer to a unique piece of context information as a document.

The simplest way to model context information (without making any assumptions about context information) is through a single dimensional model (see Fig. 1). Each document is characterized with a unique key, stored in a key-value structure and indexed by an information retrieval system.

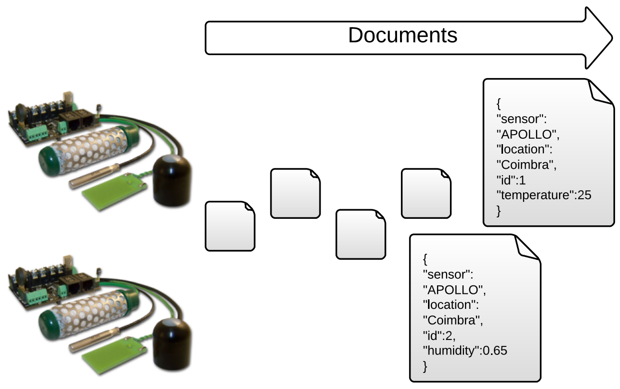


Fig. 1. Representation of a single dimensional model.

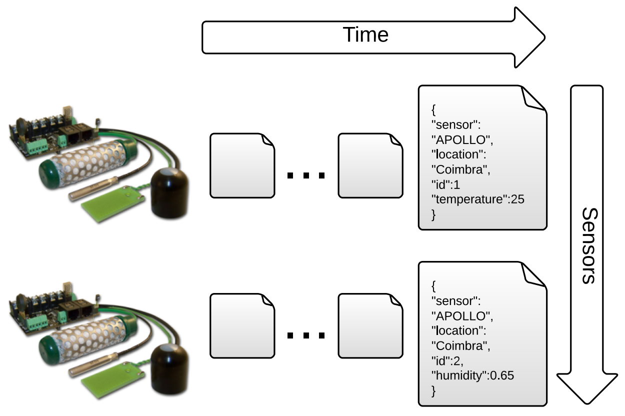
This organization model is as broad as the common definitions of context information. As a consequence it does not not take full advantage of the information retrieval system. Which can lead to poor semantic extraction, semantic ambiguity and scalability issues. Consider a scenario where some sensors published several times faster than the others. The information retrieval system is flooded by the sensors that publish at higher rates. As a consequence, the terms present on these messages are penalized (too common in the corpus). It becomes quite difficult to search information with these terms. It is not trivial to distribute the information retrieval system for several nodes. As a consequence, the performance of the context storage is related with the information retrieval dimension.

In order to minimize these drawbacks we propose a n dimensional model. The first dimension is the sensor identification. Instead of storing a data completely independently, they are organized by sensor. The platform stores all the documents, but only needs to index the sources (device/sensor). The remaining dimensions are used to select sets of data from a specific source. For the remaining of this paper we will only consider a two dimensional model (sensor identification and time). Higher dimension models only improve the selection process, do not minimize the number of sources in the information retrieval system. Without loss of generality let us consider a three dimensional model, composed by: sensor identification, time and location. A document is uniquely identified by a sensor id, a timestamp and a geographic location. It becomes possible to select documents based from a specific sensor, time and place. However, the sources indexed by the information retrieval system do not change. In short, a higher dimensional model can improve the selection in the storage component, but adds little semantic value to the information retrieval system.

It is possible to prove that a two dimensional model is as general as a single dimension model with two reasonable assumptions:

1) Each source (device/sensor) produces a continuous data stream.
2) The semantic value of the source can be extracted from a single document.

These assumption are verified empirically based on a careful analysis of context information in M2M scenarios. In the single dimensional model each individual document is treated independently. But in reality, the majority of sensors send information periodically or when a specific event is detected. This process is better modelled as a continuous data stream than a set of independent documents (assumption 1). As such context information is modelled with two dimensions (sensor identification and time) instead of a single dimension (see Fig. 2).



```
{
"sensor":
"APOLLO",
"location":
"Coimbra",
"id":1
"temperature":25
}
```

```
{
"sensor":
"APOLLO",
"location":
"Coimbra",
"id":2,
"humidity":0.65
}
```

Fig. 2. Representation of a two dimensional model.

Without making assumptions about context information the single dimensional model must use a canonical information retrieval system to provide discriminative retrieval. Each document is analysed and divided into discriminative terms. The document is analysed without taking into account their semi-structured representation. The data sent by the sensors, commonly published in semi-structured representations (e.g. XML, YAML, JSON, BSON), can be mapped into a entity-attribute-value (EAV) model [20]. The sensor is the **Entity**, what it measures are the **Attributes** and the measurements itself are the **Values**. The semantic value of the data is in the **Attributes**, the **Values** are physical measures that change over time and by itself have little semantic value. As a consequence, most of the semantic value of the stream can be taken from a single document (assumption 2).

In summary the single dimensional model is broader than the two dimensional model. Due to this, the single dimensional model it is not optimized for M2M scenarios. The two dimensional model was specifically developed for managing context information in M2M scenarios. In Section IV we compare several context storage approaches and discuss how the impact of the two organization models. We demonstrate how a organization model can improve scalability and the retrieval process.

## IV. Context storage solutions

In this section we analyse several context storage approaches. A single dimensional database can be implemented with a relational, document store and key-value database. The first two approaches (relation and document store databases respectively) are rather limited and should only be used in simple M2M scenarios. The third solution implements a singe dimensional model through a combination of a key-value database with a canonical information retrieval system [18]. Table I summarizes which requirements are fulfilled by the different approaches to context databases.

TABLE I. Requirements fulfilled by each solution.

| Databases/ Requirements | Generalize store | Discriminative retrieval | Scalability |
|---|---|---|---|
| Relational Database | Partial | Partial | Implementation dependent |
| Document Store | **Full** | Partial | Implementation dependent |
| Key-value Database | **Full** | **Full** | Partial |
| Proposed Solution | **Full** | **Full** | **Full** |

**Relational databases**, in general, are not completely suitable for storing context information. None of the key requirements for complex M2M environments are completely fulfilled. It is possible to approximate a single dimensional model with relational databases: context information can be stored in a single table, some relational databases already support full-text search and the majority of the databases support replication. Sensory information is stored as text fields (key-value approach), using a single table with three columns: the first column holds a unique identifier, the second column holds the timestamp document and the last column holds the document. The third column is then indexed by the full-text search engine.

The full-text engine, present in relational databases, is rather limited. They lack several features present in more flexible information retrieval system. They contain a simple index that contains keywords instead of primary or external keys. The search is based on keyword similarity and not concept similarity.

**Document store databases** are databases designed to store, retrieve and manage documents. In this context, documents are semi-structured data, typically encoded in XML, YAML, JSON or BSON. Documents, for this type of databases, are similar to records in a relational database. However, they are less rigid and do not follow a standard schema. Document-oriented databases typically provide a query mechanism based on information retrieval systems or custom indexes.

A document store database implies conversions of document representations, whenever the source document representation is different from the document representation. The query mechanism is also limited. Although based on information retrieval systems the indexes are commonly created automatically based on the document structure. Therefore, in order to retrieve documents it is necessary to known the structure of the documents.

As previously mentioned these approaches are rather limited. There is no clear separation between the storage and the information retrieval system. Which makes it rather difficult to add a custom EAV document analyser or define with entities must be indexed. Another issue is that these solutions scalability is closely related with their implementation and not with their approach.

The third approach combines a **key-value database** with an information retrieval system. Each individual document is treated independently, stored in the key-value database and indexed in the information retrieval system. This approach suffers from the main disadvantage of a single dimensional model solution: poor scalability. However, scalability is one of the most important requirements for context storage solutions.

Its considerably simple to scale storage through several machines. Each machine contains a portion of the information. However, the same method is not easily applied to a information retrieval system. The information retrieval system requires all the corpus to identify the most relevant and discriminative terms. A single dimension model analyses and indexes every document without taking into account its source. As the number of documents grow the performance of the index degrades rapidly.

As a counterpart to these approaches, we propose a modular context database that organizes context information in two dimensions. Our implementation combines Apache Cassandra (**tabular database**) with Apache Lucene (information retrieval system). The proposed solution has two major advantages: improved semantic extraction and scalability. Mapping sensorial data into a conceptual EAV model allow us to only extract terms with semantic value improving the semantic extraction. Taking into account that each source produces a continuous data stream, we only need to analyse some documents to semantically characterize the source. We have to store all the documents but only need to index information related to the sources and not all the individual documents (millions of documents but only some hundreds of sources).

Our proposed organization model (with two dimensions) analyses some documents per source and only indexes the source, not all individual documents. The performance of the information retrieval system depends on the number of sources and not on the number of documents. There are millions documents, but only hundreds of sources, as such the performance of the information retrieval system is only affected with the addition of new sources.

## V. Implementation

In this section we discuss important details about our solution. The context storage is divided into 3 different components as depict in Fig. 3. The storage and index components store and index context information respectively, the router communicates with the index and storage components in order to fulfil each operation. All the components communicate with each other through message passing.

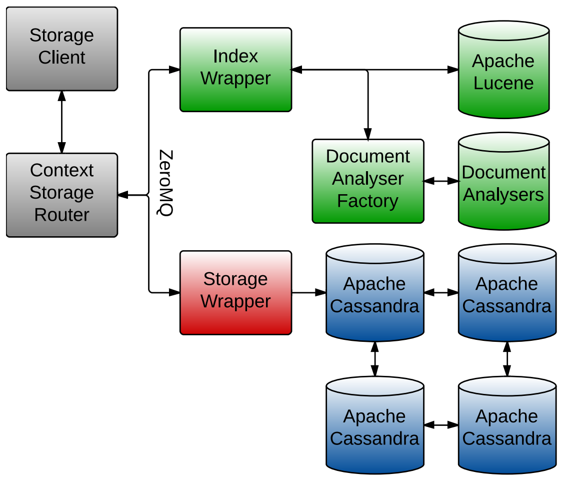


Fig. 3. Propose context storage architecture.

The components communicate with each other using the ZeroMQ[1] socket library. ZeroMQ supports several transportation methods: TCP sockets, inter-process communication and inter-thread communication. Messaging passing allows the application to be distributed through several machines and each component can be written in any programming language, without being restricted by the router component. This strategy is then specially suitable for the diversity of environments in M2M applications.

Although the modular solution in our architecture has several advantages, it can also produce sub-optimal solutions. Conceptually it is easier to devise storage and index component as two independent components. The router component (central component) decomposes each operation as a sequence of independent operations from the remaining components.

In order to decrease coupling, each component follows two design patterns: abstract factory and command. Each operation is encapsulated as a message object and each message object is in turn encapsulated as an action object. The abstract factory creates an action object from message objects, this way the translation is transparent to the component. The command design pattern makes the process of executing an action transparent to the component. Each action object implements a execute method that contains the necessary steps to execute the respective action. Another advantage of the command pattern is multitasking, each operation of the component is independent of each other, so multiple requests can be performed at the

[1]http://www.zeromq.org/

same time using different threads (as depicted in Fig. 4). The combination of abstract factory and command design pattern allows us to achieve a modular storage system with focus on parallelism instead of single request performance.

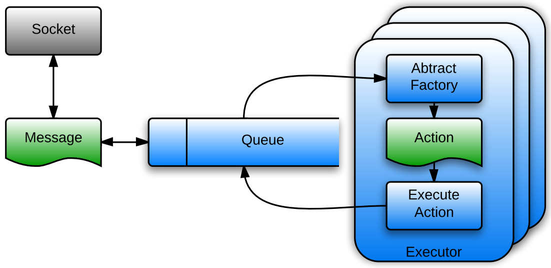


Fig. 4. General architecture of the components.

The index component is mainly an information retrieval system, responsible for indexing and searching relevant documents. It was prototyped in Java, using Apache Lucene[2] at its core. This component was developed with special attention to parallelism. The IndexWriter class was expanded to support periodical commits (safe store in the disk) with a background thread. The component also uses near-real-time search[3]. This feature allows an index changes to be visible to a new searcher with fast turnaround time. We also developed a custom document analyser that maps JSON documents into a EAV model and extract the semantic value of the attributes.

The storage component is mainly a tabular database responsible for storing all the documents. It was prototyped in Java, using Apache Cassandra [4] as its core. Apache Cassandra is one of the fastest tabular databases currently available, initially developed by Facebook and inspired by Amazon's Dynamo [21]. Cassandra is designed to handle big data workloads across multiple nodes with no single point of failure. The context information is stored in a single table with three columns: the first column holds a unique identifier, the second column holds the timestamp document and the last column holds the document.

## VI. Performance Evaluation

In Section III we analysed two context organisation models: single dimensional and two dimensional mode. As previously discussed a single dimensional model can lead to poor scalability. All the documents are indexed, even if generated from the same source. As the number of documents grows the performance of the information retrieval system degrades. Our two dimensional model only indexes the sources of data and not individual documents, optimizing the information retrieval system for M2M scenarios.

We analysed the performance of the two organization models on a smart city simulation, specifically a pothole detection scenario. It is one of the main scenarios of the APOLLO project [5]. The APOLLO project's main objective is the development of a platform that supports new scenarios in the area of M2M communications. The pothole detection scenario consist of identifying potholes on the road based on the vibrations of the vehicle. (similar use-case to project [22]). In this scenario 50 vehicles have a sensor that measures the acceleration, geographic location and the speed. The sensor sends information only when the vehicle has a speed greater than 2.5 $m/s$. After analysing the sensor's information flow we estimated that a sensor sends on average 629 documents per day.

To better understand the impact of the models in the information retrieval system it is necessary to understand how these systems work. For this evaluation we are not interested in document decomposition, term filtering or term weighting process. We are specifically interested in the model used by the information retrieval system. The majority of information retrieval use a term-document matrix to compute the similarity between documents and queries. In a term-document matrix specific type of co-occurrence matrix, each row represents a unique term and each column represents a document. The size of the co-occurrence matrix can be used as a rough estimate for the size of the index.

In the considered scenario the number of terms is constant, all the sensors send information in the same format. Fig. VI represents a sample of the information sent by the sensors, some fields are omitted for privacy reasons. On average these documents have 73 terms. Several of them are numerical values and are discarded by the standard document analyser ending up with 13 terms. Our custom document analyser (a combination of a standard document analyser with a EAV model) only extracts the attributes. The number of terms is further reduced to 11.

```
{
    "accelerometer": {
        "x": [...],
        "y": [...],
        "z": [...]
    },
    "geohash": "...",
    "id": "...",
    "latitude": 40.6325,
    "longitude": -8.6436,
    "speed": 16.81,
    "timestamp": [...],
    "version": 1
}
```

Fig. 5. Sample document sent by a sensor used in APOLLO.

For this specific scenario, and considering a single dimensional model, the index size can be estimate with the following expression: $IS(t) = sources \times rate \times terms \times t = 50 \times 629 \times 13 \times t$. $IS$ stands for index size. $sources$ are the number of sources of information (in our scenario there are 50 sources of information), $rate$ is the average number of documents sent by the sensors per day, $terms$ is the number of terms in the documents and $t$ represents the passage on time in days.

Using a two dimensional model, in this specific scenario, the index size does not depend of time. We can estimate the index size with the following expression: $IS(t) = sources \times$

[2]lucene.apache.org/core/
[3]blog.mikemccandless.com/2011/06/lucenes-near-real-time-search-is-fast.html
[4]cassandra.apache.org
[5]http://atnog.av.it.pt/projects/apollo

$terms = 50 \times 11 = 550$. The index contains the sources and not individual documents, as such from day one the index is complete. It grows only if a new source is added.

The two estimates are plotted in Fig. 6. It is visibly that with a single dimensional model, the index size grows rapidly. As a consequence the performance of the index decreases. On the other hand, with a two dimensional model, the index size is more stable. The index only grows with the addition of a new information source. In short, with a two dimensional model the index performance is limited by the number of sources and not by the number of documents.

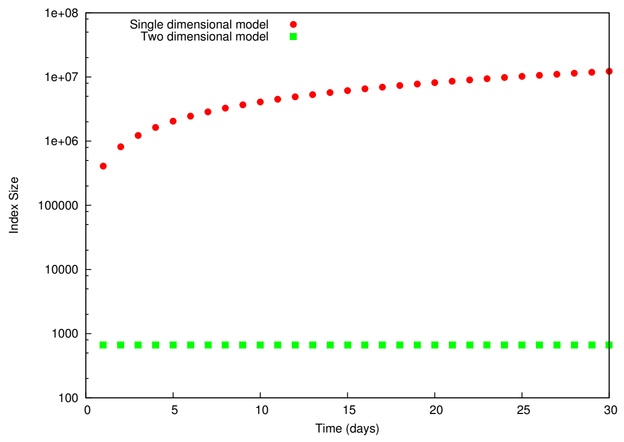


Fig. 6. Estimation of the index size.

Single dimensional models have another major drawback apart from poor scalability. The index stores all the documents regarding their source. Documents sent by the same source have similar semantic value. As a consequence, the term-document matrix has a larger amount of repeated information. Ultimately this lead to poor semantic extraction and semantic ambiguity (curse of dimensionality [23]). Taking into account our smart city scenario, consider a search by "accelerometer" where only the first 50 results are analysed. A storage solution that implements a two dimensional model returns exactly 50 sources. However, a storage solution that implements a single dimensional model returns 50 documents that might not cover the 50 relevant sources.

## VII. Discussion and Conclusions

As the number of sensors increase, it becomes increasingly difficult to store, process and analyse context information. There are countless context representation schemes, however none of them have been widely either adopted by the academia or the industry. Within this paper, we defined the basic context storage requirements, analysed the impact of context organization models and proposed a new context storage architecture for generic M2M applications

We defined the basic requirements for context storage systems, and two context organization models. The organization models were evaluated with a simulation based on a smart city scenario. Four context storage solutions were presented and analysed. The first two (based on relational and document store databases) are rather limited. The third solution (key-value database combined with a canonical information retrieval system) implements a single dimensional model to organize context information. This model leads to poor semantic extraction and scalability. We proposed a storage solution that implements a two dimensional model, minimizing the drawbacks of the precious approach. Our solution fulfils a broader set of requirements than the remaining.

## Acknowledgement

This work has been partially funded by the Portuguese Innovation Agency/National Strategic Reference Framework (AdI/QREN) under grant agreement No. 2011/021580 (APOLLO project) and by project Cloud Thinking (CENTRO-07-ST24-FEDER-002031), co-funded by QREN, "Mais Centro" program.

## References

[1] U. F. I. S. Group, "Future internet report," ICT Knowledge Transfer Network, Tech. Rep., May 2011.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proc. of the 1st international symposium on Handheld and Ubiquitous Computing*, 1999, pp. 304–307.

[3] T. Winograd, "Architectures for context," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 401–419, December 2001.

[4] M. Knappmeyer, S. L. Kiani, C. Frà, B. Moltchanov, and N. Baker, "Contextml: a light-weight context representation and context management schema," in *Proc. of the 5th IEEE international conference on Wireless pervasive computing*, 2010, pp. 367–372.

[5] M. Botts and A. Robin, "Opengis sensor model language (sensorml) implementation specification," OGC, Tech. Rep., July 2007.

[6] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *The Knowledge Engineering Review*, vol. 18, pp. 197–207, Setember 2003.

[7] M. Sabadello and D. Reed, "Oasis xri data interchange," https://www.oasis-open.org/committees/xdi/charter.php, accessed: 22-07-2013.

[8] B. Schnabel, T. Comerford, and D. Filip, "Oasis xml localisation interchange file format," https://www.oasis-open.org/committees/xdi/charter.php, accessed: 22-07-2013.

[9] B. Moltchanov, M. Knappmeyer, C. A. Licciardi, and N. Baker, "Context-aware content sharing and casting," in *Proceedings of ICIN*, 2008.

[10] T. Mota, N. Baker, B. Moltchanov, R. Ioanna, and K. Frank, "Towards pervasive smart spaces: A tale of two projects," in *Future Network & Mobile Summit 2010. The Second International Workshop on Information Quality and Quality of Service for Pervasive Computing in Conjunction with IEEE PERCOM 2010*, 2010.

[11] P. Lopes and J. L. Oliveira, "Coeus: Semantic web in a box for biomedical applications," *Journal of Biomedical Semantics*, vol. 3, no. 1, p. 11, 2012.

[12] C. Shirky, "Ontology is overrated: Categories, links, and tags," http://shirky.com/writings/ontology_overrated.html, May 2005, accessed: 22-07-2013.

[13] G. Avram, "At the crossroads of knowledge management and social software," *Electronic Journal of Knowledge Management*, vol. 4, no. 1, pp. 1–10, January 2006.

[14] T. Gruber, "Ontology of folksonomy: A mash-up of apples and oranges," *International Journal on Semantic Web and Information Systems*, vol. 3, no. 2, pp. 1–11, 2007.

[15] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. of the 2003 conference on Empirical methods in natural language processing*, 2003, pp. 216–223.

[16] R. Mihalcea and P. Tarau, "Textrank: Bringing order into texts," in *Conference on Empirical Methods in Natural Language Processing*, 2004.

[17] S. Rose, D. Engel, N. Cramer, and W. Cowley, *Automatic Keyword Extraction from Individual Documents*. John Wiley and Sons, Ltd, March 2010, pp. 1–20.

[18] M. Antunes, D. Gomes, and R. Aguiar, "Context storage for m2m scenarios," in *Proc. ICC 2014*, 2014.

[19] S. Ceri, M. Negri, and G. Pelagatti, "Horizontal data partitioning in database design," in *Proc. of the 1982 ACM SIGMOD international conference on Management of data*, 1982, pp. 128–136.

[20] P. M. Nadkarni, L. Marenco, R. Chen, E. Skoufos, G. Shepherd, and P. Miller, "Organization of heterogeneous scientific data using the eav/cr representation," *Journal of the American Medical Informatics Association*, vol. 6, no. 6, pp. 478–493, 1999.

[21] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, October 2007.

[22] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th International Conference on Mobile Systems,Applications, and Services*, 2008, pp. 29–39.

[23] R. E. Bellman, *Adaptive control processes - A guided tour*. Princeton, New Jersey, U.S.A.: Princeton University Press, 1961.