A survey of RDF store solutions

Gianfranco E. Modoni
Institute of Industrial Technologies and Automation
National Research Council
Bari, Italy
gianfranco.modoni@itia.cnr.it

Marco Sacco, Walter Terkaj
Institute of Industrial Technologies and Automation
National Research Council
Milan, Italy
{marco.sacco, walter.terkaj}@itia.cnr.it

***Abstract*—This paper analyzes the potential of Semantic Web technologies to support innovation in industrial scenarios. The study focuses in particular on RDF stores, the software components dedicated to the storage, representation and retrieval of semantic information. Starting from a literature review, a qualitative analysis is carried out considering a set of these systems. RDF stores are evaluated to find the implementations that are best suited to play the role of backbone in a software architecture sharing information between the software tools adopted by the various stakeholders. This can be achieved if the architecture overcomes the problems deriving from the lack of integration between the involved software applications, providing in this way an integrated view on relevant engineering knowledge.**

Keywords— Responsible Innovation; Interoperability; Semantic Web; RDF Stores

I. INTRODUCTION

Semantic interoperability represents a consolidated conceptual paradigm for joining heterogeneous distributed data sources into a synergistic view that facilitates the exchange of information and the use of the information that has been exchanged [1]. This is the reason why it is considered a critical success factor for optimizing business performance and maintaining competitive advantage of the manufacturing enterprises, where an inadequate semantic interoperability causes efficiency losses, whose real costs are often hidden and not easy to quantify. A study conducted in this area has estimated the cost of inadequate interoperability in the U.S. electric power industry to be $10 billion a year [2]. In particular, semantic interoperability is becoming a crucial aspect for the success of production systems while trying to reduce and master the complexity caused by the growing demand for highly complex products and the need to react to rapid changes in market demands [3]. This paradigm shift can be achieved through the effective involvement of customers, suppliers and other relevant stakeholders across the value chain, according to an approach based on the principles of Responsible Research and Innovation, realizing in this way a personalized and proactive knowledge-based production. Thus, powerful engineering tools are essential to address efficient sharing of processes and products information across the entire product lifecycle and among all actors involved. Many of these tools have been developed over recent years but their interoperability requires further studies and researches. This requirement is becoming more and more relevant to meet the new challenges posed by massive spread of Big Data [4], mobile technologies and Internet of Things (IoT) [5] also as part of manufacturing enterprises. In this context the development of robust interoperable tools is also required to combine, integrate and process large amounts of live streaming data generated from highly distributed devices.

Recent studies in the context of factories and manufacturing systems (e.g. the European projects LinkedDesign [6] and Virtual Factory Framework [7]) investigated if Semantic Web technologies [8] can be a valid solution to achieve semantic interoperability between different heterogeneous systems such as agents, services and applications. The Resource Description Framework (RDF) [9] is the standard model for data interchange on and off the Web and provides a general and flexible method to express data as lists of statements in the form of triples composed of subject-predicate-object; each list intrinsically represents a labeled, directed multi-graph, in which each triple is represented as a node (subject) – arc (predicate) – node (object) link. One of the strengths of RDF is that, in this format, any type of information can be virtually expressed. The expressivity and flexibility imply, as the flip side, the need to revise, for data expressed in such language, some classical data management problems, including efficient storage and query optimization. This is the reason why as the request of semantic applications in real scenarios are continuously growing, the need to efficiently store and retrieve RDF data is getting more and more relevant. This need has posed significant technological and engineering challenges for researchers and technologists in terms of scalable and efficient *stores*. A lot of work to improve the state of the art of these solutions has been done and several stores have been implemented. Each of the implemented stores works better under certain circumstances (e.g. certain types of data and/or queries) and may be more suitable in specific cases, depending on the requirements of the scenario.

The purpose of this paper is to present a qualitative comparison of a set of RDF store solutions. The study, based on the available literature, aims to identify a valid backbone to support large-scale semantic applications in the context of enterprises facing the challenge of realizing a more responsible and integrated management of the innovation process [10]. The requirements of collaborative and multidisciplinary environments are taken into account, since an interoperability enhancement of the involved software tools would contribute to strengthen the process innovation via an effective knowledge sharing within the organization.

The remainder of this paper is structured as follows. Section II presents the typical architecture and an updated classification of existing RDF stores based on the adopted architecture. Section III introduces the evaluation framework adopted for the comparison, whereas Section IV illustrates the results of the comparison carried out on the most widespread RDF stores and based on the previously defined criteria. Finally, Section V draws the conclusions, summarizing the major findings.

II. THE ARCHITECTURE

RDF stores are purpose-built databases for the storage and retrieval of any type of data expressed in RDF. Although the term triple store often refers to that kind of systems, the term RDF store is used in order to abstract over any type of system that is capable of handling RDF data, allowing the ingestion of serialized RDF data and the retrieval of these data later on, and providing a set of APIs to facilitate the integration with other third party applications (client applications).

Main components of the RDF store are (Fig. 1) [11]:

a) the repository (a set of files or alternatively a database).

b) the middleware in constant communication with the underlying repository.

Various database can be used as repository, ranging from one of the traditional and already tested relational database to an emerging Not Only SQL (NoSQL) database (both solutions will be discussed in detail in the following), from custom database solutions to fully supported products from specialized vendors.

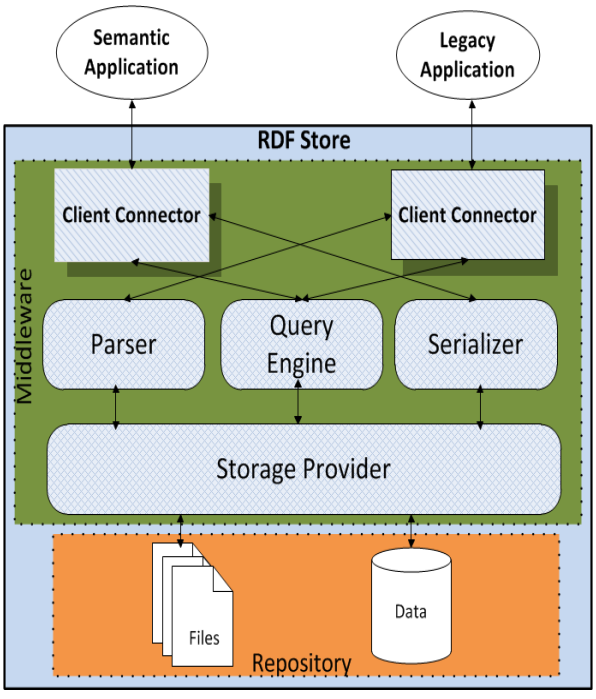

Figure 1. The architecture of an RDF Store

The middleware can be further broken down into the following components:

1) the *Storage Provider*, handling direct communication with the repository.

2) the *Query Engine*, providing the capability to retrieve information from the repository according to structured queries, formulated in one of the defined RDF query languages. SPARQL Protocol and RDF Query Language (SPARQL) [12], representing W3C recommendation, is emerging as the de facto RDF query language; it defines a protocol for sending queries from client applications to the component that can execute the queries, the SPARQL endpoint, and for retrieving the requested results.

3) the *Parser/Serializer*. RDF has a number of serializations, including the W3C standard serialization in XML. An RDF parser reads text in one (or more) of these formats and interprets it as triples in the RDF data model. An RDF serializer does the reverse; it takes a set of triples and creates a file or a memory buffer that expresses that content in one of the serialization forms.

4) the *Client Connectors*, available in one or more programming languages, for programmatically accessing and manipulating the contents of the store from any legacy or semantic client applications.

It is important to note that the implementation of whatever of the above mentioned components is usually transparent to any client application, because this latter interacts with RDF store only through standardized interfaces of the access component. The same goes for the repository, for which it is possible to use different types of databases interchangeably, without modifying the client application that uses the RDF store.

Current RDF stores can be grouped in three major categories: database based stores, native stores, and hybrid stores. Database based stores (e.g. Oracle 12c[1], Redland MySql[2]) have been built on top of existing commercial database engines; native stores (e.g. AllegroGraph[3], OWLIM[4]) have been built as database engines from scratch, whereas hybrid stores (e.g. Virtuoso[5], Sesame[6]) support both architectural styles (native and DBMS-backed).

In the past, several stores have been proposed and implemented based on traditional widely tested relational databases. Initially this approach allowed large and powerful solutions to be constructed with little programming effort. However, the flexible model of RDF is poorly suited to traditional relational storage models that, for reasons of efficiency, rely on well-defined structural expectations.

[1] http://www.oracle.com/technetwork/database/enterprise-edition/overview/index.html
[2] http://librdf.org/
[3] http://www.franz.com/products/allegrograph/
[4] http://www.ontotext.com/owlim/
[5] http://virtuoso.openlinksw.com
[6] http://www.openrdf.org/

Therefore, the recent trend for managing RDF data has moved away from relational approach to ones better suited for exploiting the graph form of RDF data, thus following the NoSQL paradigm. The latter provides a mechanism for storage and retrieval of data that do not use a relational data model; the reasons for choosing this approach include simplicity of design and horizontal scaling. There are various models of NoSQL databases; a few of these with their prototypes are:

- Column: HBase, Cassandra;
- Document: MongoDB, MarkLogic;
- Key-value: Dynamo, Project Voldemort;
- Graph: Neo4J, AllegroGraph.

A graph model based database, not depending on a rigid schema, fits more properly to the flexible structure of RDF data; for this reason it can scale more naturally to large datasets, mitigating some of the performance problems of relational model–based stores.

III. THE EVALUATION FRAMEWORK

This section introduces the evaluation framework adopted for the qualitative analysis of a set of well-known RDF stores.

The overall aim of this study is to find an effective RDF store in the role of backbone of an integrated software solution which supports enterprises to realize an approach based on public debate, broad stakeholder involvement and early societal intervention in research and innovation [10]. One of the main needs of a such solution is the capability to manage huge amount of near real-time data, also in the form of Big data. In fact this allows to collect and gather billions of real-time bytes of data on its customers, resources, or products, that are then processed instantaneously to optimize resource utilization or customer satisfaction. However, the capability to handle intensive data is only the first step towards the consolidation of an enterprise model where actors are involved to become co-responsible for the innovation process, for example participating in the definition of societal desirable products. In order to realize such a scenario, other key features are required including the capability to manage multiple users with different levels of access to resources and the availability of a version management system to retain historical versions of data [13].

A. Functional and non-functional characteristics

This subsection identifies a set of dimensions, selected from functional and non-functional characteristics, that can be used as criteria to perform the evaluation of a set of RDF stores: handling streaming data, security, versioning, handling binary data. As seen in Section 1, the architecture of RDF stores is composed of various components and some of the selected criteria are applicable only to a subset of these components.

Handling streaming data. In modern enterprise the amount of live streaming data produced by highly distributed devices, such as sensors and actuators, is continuously growing. These data are produced as continuous streams usually at high rates, e.g. once per millisecond or even higher. The heterogeneity of these data, their multiplicity and their rate of change over time lead, in real-time systems, to complex issues related to both the overall performance and the interpretation of the data. In reference to this last issue, the lack of integration between different live streaming data resources often isolates relevant data streams and raise the problem of "too much (streaming) data but not enough (tools to gain and derive) knowledge" [14, 15]. In order to contribute to fill this gap, it is considered crucial to investigate if Semantic Technologies (and in particular RDF stores) are mature to support streaming data.

Security. In a collaborative scenario the data access control is an essential requirement since several users can read, write, and share the stored information. For this reason the implemented RDF stores must have the ability to ensure the privacy of the users, allowing for example to manage their current private experiments before sharing interesting findings with some collaborators and eventually later with the entire community. One of the possible approach to manage security in RDF store is applying access controls at graph level or, through a finest grained security policy, at triple level, allowing software application architects and then involved stakeholders to determine how to partition data appropriately. In an effective system, access privileges are not granted to single users; instead they are granted to a set of permissions (roles). Thus, users' operations are governed by the roles to which they belong. This provides the ability to handle any situation, from a simple role containing a single person up to a hierarchy of users and roles that represents the various groups and subgroups that characterize a complex organization.

Versioning. In a collaborative design and production environment, the capability of versioning is essential in ensuring consistency among different versions of the stored models, allowing the merging of the efforts of people involved in product design, factory design, production planning, manufacturing operations and execution, etc. Thus, it is a crucial task to implement a functionality to keep track of multiple versions of a complex RDF data model expressed, in turn, under the form of a complex hierarchy of RDF data models which are also versioned due to their evolutions. Moreover, this complexity increases in a concurrent engineering design and production environment where a large number of stakeholders may be involved across the entire product lifecycle applying continuous changes by several interactions. In this scenario, the versioning capability enables to work concurrently on different versions of RDF data, to come back to a previous stable state whenever changes introduce some problems and to compare versions of models and highlight the differences, helping in finding changes, even if those have occurred in an uncontrolled order [16].

Handling binary data. In the context of manufacturing enterprises, big data scenarios involve massive collections of binary file, which are instrumental for the correct representation of the information (graphic files, process data, etc.). There are pros and cons in storing binary data in the RDF store. The advantages are:

- A uniform access to data and metadata;
- It is easier to keep data and metadata synchronized;

- It is easier the implementation of versioning feature.

Alternatively, it is possible to store binary data outside from the store, e.g. coupling the RDF store with a Content Management System (CMS)-like system. The list below summarizes briefly the advantages of this approach:

- The development and the debugging are easier;
- The data management is easier (e.g. for backups);
- It is faster because it keeps the "big blobs" out of the RDF store;
- It avoids problems with special characters contained in binary data.

B. *Technical characteristics*

Apart from the functional and non-functional features, RDF stores are also rated according to relevant technical characteristics and other general information; the following list summarizes the most important.

License. The information about the usage of the store. There are two main categories of licenses for these technologies: commercial and open source under various licenses (Apache License, GNU GPL, etc.).

Operating system. The operating system (e.g. Linux, Windows, Mac OS X) on which the RDF store runs.

Implementation language. The programming language that developers have used to implement the store.

Repository mechanism. The category of membership according to the taxonomy introduced in section II (database based, native, hybrid); it includes information about the type of used repository (Relational, NoSql, etc.).

Client connector languages. The programming languages (C#, C, Java, PHP, etc) in which client connectors have been implemented. It is a criterion that can be used to judge the ability to collaborate with other applications.

Query Languages. The query language(s) supported.

Latest Release Date. When the latest release of the RDF store was released. If the date is not recent, the product may no longer be supported.

IV. ANALISYS OF RDF STORES

A. *Preselection of RDF stores to be evaluated*

As the storing, accessing, and processing of large amount of data represents an essential requirement in order to satisfy the needs to search, analyze and visualize these data as information, the capability to manage a large number of triples (greater than 2 billions) has been chosen as filter criterion for selecting a set of stores to be evaluated. Based on this filter, it has been selected a list among the most well-known RDF stores that satisfy this requirement, excluding all technologies which are discontinued or in a too early state of development.

The list is reported below; for each item it is also reported a brief overview.

AllegroGraph is a closed source RDF store developed by Franz Inc. in Common Lisp. Currently it is in use in various open source and commercial projects. It is the storage component for the TwitLogic project that is bringing the Semantic Web to Twitter data.

Bigdata(R)[7] is a platform developed by SYSTAP LLC, that can be deployed as an embedded database, a standalone server, a replication cluster, and as a horizontally-shared federation of services similar to Google's bigtable, or Cassandra.

Apache Marmotta[8] is a platform implemented as a Service-Oriented Architecture (SOA). Currently, the following backends are available: KiWi Triple Store, Sesame Native and BigData (experimental).

OpenLink Virtuoso is developed by OpenLink Software as a database engine hybrid that combines the functionality of a traditional RDBMS, virtual database, RDF, XML in a single system. It is used to manage data exposed by dbpedia.org.

Oracle 12c supports RDF data management with his component Oracle Spatial and Graph RDF Semantic Graph. It is a commercial product, which is free of charge for non-commercial purposes. There are some examples of use in integrated bioinformatics, health care informatics, finance, web social network, and content management.

Stardog[9] is a graph database based on pure Java storage and designed for mission-critical applications.

B. *Comparison of RDF stores*

This subsection presents the results of the comparison based on the previously adopted evaluation framework and carried out on the set of preselected RDF stores. To facilitate the analysis of the technical characteristics, a synopsis of main outcomes of this study is reported in Table I.

In the following it is presented, for each criterion included in the adopted framework, the evaluation of the selected RDF stores.

Handling streaming data. Very little information is available at this time about how the surveyed RDF stores act when they handle streaming data. *AllegroGraph* has been involved in an experimental evaluation of an architecture serving as a real-world example of a high-performance RDF streaming application that sends high-throughput updates to remote store in an Internet-scale distributed environment [17]. There are examples of *Stardog*'s use as backbone of software application for near real-time representation of situational knowledge acquired, also at high frequencies, from heterogeneous sensor network data [18].

[7] http://www.systap.com/bigdata.htm
[8] http://marmotta.apache.org/
[9] http://stardog.com/

TABLE I. EVALUATION SYNOPSIS OF A SET OF TECHNICAL CHARACTERISTICS

	Technical characteristics						
	License	***Operating System***	***Language***	***Repository Mechanism***	***Client Connectors***	***Query Language***	***Latest Release Date***
AllegroGraph	Commercial, Free	Windows, Linux, Mac	Lisp	Native (Graph)	Java, Python, Lisp, Ruby, Perl, C#	SPARQL	Feb 2014
Bigdata	Commercial, GNU GPL	Linux / Unix	Java	Hybrid	Java	SPARQL	Dec 2013
Marmotta	Apache2	Windows, Linux, Mac	Java	Hybrid	Java	SPARQL	Jan 2014
Openlink Virtuoso	Commercial, GNU GPL	Windows. Linux, Mac	C	Hybrid (object/relational)	Drivers for ODBC, JDBC, ADO.NET and OLE DB	SPARQL, SQL	Feb 2014
Oracle 12c	Commercial, Free	Windows, Linux, Solaris	Java	DBMS backed	Connectors for SQL and Java-based applications	SPARQL, SQL	July 2013
Stardog	Commercial, Free	Windows. Linux, Mac	Java	Native (Graph)	Java, Ruby, Python, .Net	SPARQL	Feb 2014

To understand how an RDF store is able to cope with huge flows of near real-time information coming from many sources, it is necessary to analyze his performance profile in such a context and verify if it satisfies the expectations.

Several metrics can be used to facilitate a standard and systematic evaluation of the RDF stores performance when dealing with huge flows of near real-time information coming from many sources; two of the most commonly used metrics are query duration and load time. Query duration is the amount of time taken to return the result set for a specific query. Load time is the amount of time taken to add some information to the store, including any overhead occurring as part of this operation. Other useful metrics include disk space requirements, memory footprint and deletion duration.

Based on the above metrics, several benchmarks (e.g. LUBM [19], Berlin Benchmark [20], etc.) have been formalized and published. However, they do not always allow a rigorous evaluation and, moreover, do not fully satisfy all the successful requirements of a good benchmark (i.e. relevant, repeatable, fair, verifiable and economical) [21]. Therefore, a definitive evaluation of RDF store performance is still missing. Many expectations are placed on the ongoing EU project LDBC (Linked Data Benchmark Council) [21] that brings together a community of academic researchers and industry, whose main objective is the development of open source and industrial grade benchmarks for graph and RDF databases, leading to an easier comparison of the different technologies also in the context of scenarios managing massive amounts of streaming data.

Security. *Virtuoso* is based on a graph-level role-based security policy; thus, users can be granted permissions to specific graphs, either read-write or read-only.

Stardog's security model is also based on role-based access control at graph-level: in this case users can be granted permissions over resources to which access is to be controlled. Moreover, in order to provide better data protection, it supports extensible authentication via both internally stored user information and external mechanisms (e.g. LDAP) and in-flight encryption of credentials and payload data via HTTPS.

The component Enterprise Security and Management (ESM) provides *AllegroGraph* the mission critical functionality that organizations need to support 24 hours operations, including a transport layer security to and from client applications, which also have the ability to send and receive encrypted requests, and data access control at the triple level security. Moreover, *AllegroGraph* introduces Triple/Quad Level Security Filters, which can prevent access (both read and write) to triples with a specified value for subject, predicate, object and/or graph, as it is possible to specify which values should be allowed or disallowed.

Oracle 12c has the default access control at the graph level that allows the owner of a graph to grant appropriate privileges to other users. However, for applications with stringent security requirements, it is possible to enforce a fine-grained access control mechanism allowing security administrators to define policies that eventually restrict a user's access to triples that involve instances of a specific RDF class or property. *Oracle 12c* supports also standard encryption that allows administrators to choose which data to encrypt and which standard encryption algorithm to use.

Marmotta has an integrated security module which implements the authentication and authorization mechanism based on the access control list (ACL), that specify which users are granted access to specific objects and what operations are allowed.

Finally, *Bigdata* does not support triple level security. In general, very little information is available about how this RDF store implements the security mechanism.

Versioning. *AllegroGraph*, *Stardog* and *Virtuoso* do not have a built-in versioning mechanism. In such cases, on top of these stores, external components (e.g. Graph Versioning System) can be deployed in order to keep track of the RDF data contained in the underlying repository. However, this solution is less efficient compared to the case in which the versioning functionality is internal to the RDF store.

Versioning is available in *Marmotta* on top of the supported backend KiWi triple store. Currently, it allows logging of changes applied to the repository resources, including the source (origin) of the saved triples. Moreover, it offers the capability to create snapshots of the repository that have reached an acceptable level of quality and stability. These snapshots can be later referenced while updating the repository with new data. Instead reverting changes has not yet been implemented and will be added later.

As for many applications the unlimited history is not required, *Bigdata* provides a configurable policy to retain historical data for a given period. At any point of this interval, it is possible to request a consistent view of the database and the corresponding data can be read.

Most of the stores surveyed do not currently implement the versioning feature in an exhaustive and satisfactory manner. However, much of them are making great efforts to fill this gap, despite the difficulties linked to several constraints imposed on versioning algorithms by RDF model.

Oracle 12c is among the technologies that implement the versioning capability in a more comprehensive way, thanks to his component Workspace Manager which provides a mechanism where the granularity (unit) of versioning is the graph model. It supports a collaborative development project where a team can share access to a collection of insertions and updates, allowing also control to the applied operations.

Handling binary data. Most of the current implemented stores do not have an optimized BLOB (Binary Large Object) storage, same like in the most of relational databases.

Though *Stardog*, *Marmotta* and *Bigdata* can store any legal RDF value, they are not best used as a blob stores.

AllegroGraph supports CLOB (Character Large Object), useful to handle huge strings, but it cannot directly support BLOB.

Virtuoso is capable of storing binary data storage, exploiting the built in WebDAV repository, which can host static and dynamic web content.

Finally, *Oracle 12c* support large volume of data under form of BLOB and it can hold up to 4 GB of data for a single resource, thus ensuring an effective storing of digitized information (e.g. images, audio, video).

V. CONCLUSIONS

As a result of this survey it can be said that various implementations of RDF store are suitable to be used as backbone of semantic applications that need to store and process large amount of RDF data in a safe and reliable manner. Moreover, most of them usually provide support for essential features such as the capability to guarantee data protection, information privacy and security. However, the majority of the surveyed RDF stores do not yet support versioning and handling streaming data in an effective way. As from a responsible industrial perspective they embody essential features, their lack is an important gap representing the most pressing technological barrier that researchers and technicians have to overcome in the next future.

Some key characteristics of an RDF store have not been considered in this survey. One of the most important is the support to inference, i.e. the capability to derive new knowledge directly from information already available. This is a crucial functionality and for this reason it has set the objective to conduct a new survey considering it as evaluation criterion, also extending the set of surveyed technologies.

Finally, a quantitative analysis of RDF stores performance has not been carried out in this study and it will require further comparative evaluations. The aim is to face this challenge in future work, relying on new benchmarks, currently under study.

ACKNOWLEDGMENT

The research reported in this paper has been partially funded by the MIUR under the Italian flagship project “La Fabbrica del Futuro”, Subproject 2, research project PRO2EVO Product and Process Co-Evolution Management via Modular Pallet configuration, and by the European Union 7th FP (FP7/2007–2013) under the grant agreement No: 314156, Engineering Apps for advanced Manufacturing Engineering (Apps4aME), and by Regione Lombardia under the project “Fabbrica Intelligente per la Deproduzione Avanzata e Sostenibile” (FIDEAS).

REFERENCES

[1] IEEE Standard Computer Dictionary, "A Compilation of IEEE Standard Computer Glossaries", IEEE, 1990.

[2] "Financial Benefits of Interoperability: How Interoperability in the Electric Power Industry will Benefit Stakeholders Financially", Prepared for the GridWide September 2009 Architecture Council by Harbor Research, 2010.

[3] W. Terkaj, T. Tolio, and A. Valente, “Designing Manufacturing Flexibility in Dynamic Production Contexts”, Design of Flexible Production Systems, Springer, Ch. 1:1–18, 2009.

[4] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Hung Byers, "Big data: The next frontier for innovation, competition, and productivity", McKinsey Global Institute, 2011.

[5] S. Haller, S. Karnouskos, and C. Schroth, "The Internet of Things in an Enterprise Context", in J. Domingue, D. Fensel und P. Traverso (Eds.),

First Future Internet Symposium - FIS 2008, LNCS 5468, Springer Verlag 2009, pp. 14-28, 2009.

[6] A. Milicic, A. Perdikakis, S. El Kadiri, D. Kiritsis, S. Terzi, P. Fiordi, and S. Sadocco, "Specialization of a Fundamental Ontology for Manufacturing Product Lifecycle Applications: A Case Study for Lifecycle Cost Assessment", OTM Workshops 2012: 69-72, 2012.

[7] K. Botond, W. Terkaj, and M. Sacco, "Semantic Virtual Factory supporting interoperable modelling and evaluation of production systems", CIRP Annals - Manufacturing Technology 2013; 62(1):443-446, 2013.

[8] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", Scientific American, 284(5):34-43, 2001.

[9] G. Klyne, and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax", (W3C Recommendation 10 February 2004), World Wide Web Consortium, 2004.

[10] R. von Schomberg, "Towards Responsible Research and Innovation in the Information and Communication Technologies and Security Technologies Fields", A Report from the European Commission Service, Publications Office of the European Union, 2011.

[11] A. Hertel, J. Broekstra, and H. Stuckenschmidt, "RDF Storage and Retrieval Systems", On-line, 2008.

[12] W3C, "SPARQL Query Language for RDF", W3C Semantic Web Activity RDF Data Access Working Group, 2008.

[13] G. Ghielmini, P. Pedrazzoli, D. Rovere, W. Terkaj, C.R. Boër, G. Dal Maso, F. Milella, and M. Sacco, "Virtual Factory Manager for semantic data handling", CIRP Journal of Manufacturing Science and Technology, 6(4):281-291, 2013.

[14] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic Sensor Web", IEEE Internet Computing, 12(4):78-83, 2008.

[15] Y. Zhang, M.D. Pham, O. Corcho, and J.P. Calbimonte: "SRBench: A Streaming RDF/SPARQL Benchmark", In International Semantic Web Conference (ISWC 2012), Boston (USA) , 641-657, 2012.

[16] V. Papavassiliou, G. Flouris, I. Fundulaki, D. Kotzinos, and V. Christophides, "On Detecting High-Level Changes in RDF/S KBs", In Proc. of ISWC, 2009.

[17] S. Groppe, J. Groppe, D. Kukulenz, and V. Linnemann, "A SPARQL Engine for Streaming RDF Data", Signal-Image Technologies and Internet-Based System, SITIS '07, 2007.

[18] M. Stocker, E. Baranizadeh, A. Hamed, M. Rönkkö, A. Virtanen, A. Laaksonen, H. Portin, M. Komppula, and M. Kolehmainen, "Acquisition and Representation of Knowledge for Atmospheric New Particle Formation", ISESS 2013, 2013.

[19] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for OWL knowledge base systems", Web Semantics: Science, Services and Agents on the World Wide Web, Volume 3, Issues 2-3, Pages 158-182, 2005.

[20] C. Bizer, and A. Schultz, "The Berlin Sparql Benchmark", International Journal On Semantic Web and Information Systems, 2009.

[21] P. Boncz, I. Fundulaki, A. Gubichev, J. Larriba-Pey, and T. Neumann, "The linked data benchmark council project", Datenbank Spektrum 13(2):121-129, 2013.