Semantic Business Process Repository

Zhilei Ma[1], Branimir Wetzstein[1], Darko Anicic[2], Stijn Heymans[2]

[1]Institute of Architecture of Application Systems (IAAS)
University of Stuttgart, Germany
{firstname.lastname}@iaas.uni-stuttgart.de
[2]Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria
{firstname.lastname}@deri.org

Abstract. Semantic Business Process Management (SBPM) utilizes semantic technologies to achieve more automation throughout the BPM lifecycle. An integral part of the SBPM infrastructure is a Semantic Business Process Repository (SBPR), which is used for storage and management of business process modeling artifacts. As in SBPM business process models are based on process ontologies, the SBPR has additional requirements towards support of reasoning and querying capabilities. In this paper, we first identify the functionalities the SBPR has to provide. We then evaluate different approaches on how process models can be stored and queried efficiently by taking the semantic information into account. Finally, we present the overall architecture for the SBPR.

Keywords: Business Process Management (BPM), Business Process Repository, Semantic Business Process Management (SBPM), Ontologies, Reasoning

1 Introduction

The globalization of the economy and the ongoing change of the market situation challenge corporations to adapt their business processes in an agile manner to satisfy the emerging requirements on the market and stay competitive against their competitors. Business Process Management (BPM) is the approach to manage the execution of IT-supported business processes from a business expert's point of view rather than from a technical perspective [SF03]. However, currently businesses have still very incomplete knowledge of and very incomplete and delayed control over their process spaces. Semantic Business Process Management (SBPM) extends the BPM approach by adopting semantic web and semantic web service technologies to bridge the gap between business and IT worlds [HLD+05].

In both BPM and SBPM business processes play a central role. As business processes manifest the business knowledge and logics of a corporation and normally more than one person or organization with different expertise and in different geographic locations are involved in management of business processes, it is necessary to establish a Business Process Repository (BPR) within the corporation for effective sharing of valuable business knowledge. Furthermore, business users tend to reuse existing business process artifacts during process modeling, so that they are able to adapt the

business processes in a more agile manner. However, as the number of business processes increases, it is difficult for them to manage the process models by themselves and to find the required business process information effectively. A BPR helps business users by providing a systematic way to manage and obtain information on business processes.

In SBPM business process models are based on process ontologies and make use of other ontologies, such as organizational ontology or semantic web service ontology [HR07]. The BPR has to be able to cope with these ontological descriptions when storing and retrieving process models, and in particular support efficient querying and reasoning capabilities based on the ontology formalism used. In order to distinguish from traditional BPR technology, we call this kind of repository a Semantic Business Process Repository (SBPR).

In this paper, we first analyze the functional requirements on the SBPR. We describe what kind of functionality the SBPR should offer to its clients, which is primarily a process modeling tool. We then compare different approaches for data storage and querying based on the ontological descriptions. The comparison is based on the expressiveness of the query language, the scalability of the query processing and the effort for the integration of the query processing with the underlying data storage. We then finally describe the overall architecture of the SBPR.

2 Requirements Analysis

In general, a repository is a shared database of information about engineered artifacts produced or used by an enterprise [BD94]. In SBPM, these artifacts are semantic business process models (process models for short).

Process models are modeled by business users with help of a process modeling tool. To support process modeling, the SBPR has to provide standard functionality of a Database Management System, such as storage of new process models, update, retrieval or deletion of existing process models, transaction support for manipulation of process models and query capability. The query capability enables business users or client applications to search process models in the SBPR based on the criteria specified. We classify the queries into two categories. The first category of queries can be answered based on the artifacts explicitly stored in the SBPR. This kind of queries is of the same kind as the queries that traditional database systems can process. The second category of queries is “semantic queries”, which can only be processed, when the ontological knowledge of the process models is taken into account.

The modeling of business processes can be a time-consuming task. It may take days or even months for business users to finish modeling a given business process. Therefore, treating the entire modeling activity related to a process model as a single transaction is impractical. The SBPR has to provide check-in and check-out operations, that support long running interactions, enable disconnected mode of interaction with the SBPR, and are executed as separate short transactions. In this case the modeling tool works in a disconnected mode regarding the SBPR. The process model in the SBPR is locked when the modeling tool obtains it (check-out), so that no other users can modify the process model in the SBPR in the meantime. After the modeling

work has been done, the process model is updated in the SBPR and any locks that have been held for the process model are released (check-in). Please note that the locking mechanism refers only to the locking of the process models in the SBPR. The process ontologies, that are stored separately in an ontology store and have been referenced by the process models, are not locked simultaneously. Furthermore, in a distributed modeling environment several business users may work on the same process model simultaneously. A fine-grained locking of elements in a process model enables different business users to lock only the part of the process model, they are working on, thus avoiding producing inconsistent process models.

Process models may undergo a series of modifications undertaken by business users. The series of modification is called change history of the process model. The SBPR represents the change history as versions. A version is a snapshot of a process model at a certain point in its change history [BD94]. In certain industry sectors corporations must record all the change histories of their process models for government auditing or for some legal requirements. From the modeling perspective it is meaningful to keep process models in different versions, so that business users can simply go back to an old version and develop the process model from the old version further. Due to these reasons the SBPR has to provide also versioning functionality, so that the change history of process models can be documented.

3 Comparison of Storage Mechanisms

As storing and querying process models are the main requirements for the SBPR, we evaluate in this section several options for storage mechanism and their query capabilities.

A process model is an instance of a process ontology. Process ontologies which are developed in the SUPER project [SUPER, HR07] include the Business Process Modeling Ontology (BPMO); the semantic Business Process Modeling Notation ontology (sBPMN), which is an ontological version of Business Process Modeling Notation (BPMN) [BPMN06]; the semantic Event Process Chain ontology (sEPC), which is an ontological version of Event Process Chain (EPC) [KNS92]; the semantic Business Process Execution Language ontology (sBPEL), which is a ontological version of Business Process Execution Language (BPEL) [ACG+05]. These ontologies are described using the ontology-formalism Web Service Modeling Language (WSML) [WSML05]. There are 5 variants of WSML: WSML-Core, WSML-DL, WSML-Flight, WSML-Rule, and WSML-Full, differing in logical expressiveness and underlying language paradigm [WSML05]. The ontologies considered in this paper are formalized using WSML-Flight, which is a compromise between the allowed expressiveness and the reasoning capability of the ontology language. In the following, we thus assume that a process model is an instance of a process ontology, which is specified in WSML-Flight.

For each option we take into account the expressiveness of the query language, the scalability of the query processing and the effort for the integration of the query processing with the underlying data storage. Scalability is a rather fuzzy term. In the context of SBPR we define it in respect to reasoning scalability. Reasoning is used to

infer conclusions that are not explicitly stated but are required by or consistent with a known set of data (cf. [PT04]). A system or a framework is scalable if enlarging the data-set, which in our context is the set of actual process models that described using ontologies, leads to a performance loss that is tolerable. More formally, one could say that reasoning is scalable if augmenting the input size of the problem, which in this case refers to the ontologies plus the instance data of the ontologies, leads at most to a polynomial increase of the time in which reasoning can be performed. With regards to the reasoning capability we consider two options, namely the storage mechanism with or without reasoning capability.

3.1 Option 1: Without Reasoning Capability

For storage mechanisms without reasoning capability we considered Relational Database Management System (RDBMS) and RDF store, which have been widely adopted.

Queries against RDBMS are normally performed using the Structured Query Language (SQL). SQL is based on both the relational algebra and the tuple relational calculus [SKS06]. It is a powerful language, which, however, still has some limitations. For example, a simple query such as:

Find all supervisors of the employee John Smith

requires computation of transitive closures on the personnel hierarchy. It is known that transitive closure cannot be expressed using relational algebra [LL01, AHV95]. In SQL, one can express transitive closures using *WITH RECURSIVE* to create recursive views, which in general is a very expensive operation. Furthermore the "supervisor" relationship must be stored explicitly in the database system. As SQL can only express queries that target explicitly stored data, it has no capability to take into account implicit data, which can be derived from the instances of the ontologies based on the axioms specified there. Thus, SQL is not sufficient for the requirements on query processing of the SBPR.

[BKK06] defined a RDF representation of WSML, which allows storing WSML data in a RDF store. RDF store is a framework providing support for the RDF Schema [RDFS04] based inference and querying, which uses a relational database system as the underlying storage for the RDF data [RDF04]. In this section, we only consider RDF stores without an integrated third-party inference engine or reasoner. Inference here refers to the RDFS entailments supported by the RDFS semantics. There are already several reference implementations of RDF stores like Sesame[1]. Inference in such RDF stores is normally based on the RDF schema, which provides only a restricted number of constructs to describe the relationships between resources and properties, such as rdfs:subClassOf, rdfs:subPropertyOf. The query processing of RDF stores is based on special query languages for RDF data, such as Simple Protocol and RDF Query Language (SPARQL) or Sesame RDF Query Language (SeRQL). Using these query languages, one cannot express transitivity or transitive closure. Furthermore, these query languages take only into account explicitly stored data. The

[1] http://www.openrdf.org/index.jsp

implicit data can be derived by the inference capability. However, the inference capability is very limited in RDF stores.

3.2 Option 2: With Reasoning Capability

Jena[2] 2 is another RDF store, which support not only native entailment of RDFS semantics but also third-party inference engines or reasoners. The primary use of a plugged-in inference engine is to support the use of more expressive languages such as OWL which allow additional facts to be inferred from instance data and class descriptions [JENA2]. The default OWL reasoner in Jena can only perform reasoning on a subset of OWL semantics. To provide complete support of OWL DL reasoning one can use external OWL DL reasoner such as Pellet[3], Racer[4] or FaCT[5]. Jena can handle OWL DL, but there is only a partial bi-directional mapping defined between WSML-Core and OWL DL, which is not sufficient to fulfill the requirements of SBPR that requires WSML-Flight.

Besides Jena, OWLIM [OWLIM] is another implementation of a RDF store with reasoning capability. OWLIM is a high performance Storage and Inference Layer (SAIL) for Sesame[6], which performs OWL Description Logic Programs (DLP) [GHV+03] reasoning, based on forward-chaining of entailment rules [KOM05]. As argued in [KOM05], OWLIM can query the Knowledge Base (KB) of 10 million statements with an upload and storage speed of about 3000 statements per second [OWLIM]. In more detail, querying is done by materializing the KB, i.e., for every update to the KB, the inference closure of the program is computed: all conclusions that can be recursively obtained by applying Process Ontology rules, given certain instance data (process models), are computed. This approach has the advantage that querying or other reasoning tasks are performed fast because the reasoning was done beforehand. Moreover, one could store the inference closure in the persistent storage, effectively using optimization methods for storage. The approach taken in OWLIM shows, that taking into account ontologies does not need to lead to a significant performance loss per se. Nonetheless, the approach has some disadvantages.

OWLIM provides support for a fraction of OWL, close to OWL DLP and OWL-Horst [TH05], which can be mapped to WSML and vice versa. However, the expressiveness of OWL DLP corresponds to WSML-Core. OWL-Horst is more powerful than WSML-Core, but it is still not as powerful as WSML-Flight. Therefore, the expressiveness is not adequate. As we already discussed, the reasoning in OWLIM takes the forward-chaining approach. Forward-chaining means that the reasoner starts from the facts that are already known and infers new knowledge in an inductive fashion. The result of forward-chaining can be stored for reuse. This enables efficient query answering, because all facts needed for the query processing are already available in the data storage. On the other hand, the disadvantage of this approach is that update and delete operations are very expensive. Newly added or updated data leads to a re-

[2] http://jena.sourceforge.net/index.html
[3] http://pellet.owldl.com/
[4] http://www.racer-systems.com/
[5] http://www.cs.man.ac.uk/~horrocks/FaCT/
[6] http://www.openrdf.org/index.jsp

computing of the inference closure in the SBPR. Removal of process models is even more problematic, as facts from the inference closure that were introduced by this removed process models have also to be removed from the SBPR, which could lead to more removal operations. In the worst case this could lead to a recalculation of a large part of the inference closure. However, the removal of process models from the SBPR seems to be an action that is less common. The OWLIM approach also relies heavily on the fact that the semantics of OWL DLP and extensions towards OWL Lite are monotonic. The monotonic semantics allows for incremental additions to the SBPR, i.e. one can extend the current inference closure with new inferences. In the presence of non-monotonism, e.g., negation as failure as for example in WSML-Flight [BLP+06], such an incremental approach no longer works, as adding knowledge may prohibit previously made deductions.

IRIS (Integrated Rule Inference System)[7] is an inference engine, which together with the WSML2Reasoner framework[8], supports query answering for WSML-Core and WSML-Flight. In essence, it is a Datalog engine extended with stratified negation[9]. The system implements different deductive database algorithms and evaluation techniques. IRIS allows different data types to be used in semantic descriptions according the XML Schema specification and offers a number of built-in predicates. Functionality for constructing complex data types using primitive ones is also provided.

The translation from a WSML ontology description to Datalog is conducted using the WSML2Reasoner component. This framework combines various validation, normalization and transformation functionalities which are essential to the translation of WSML ontology descriptions to set of predicates and rules. Further on, rules are translated to expressions of relational algebra and computed using the set of operations of relational algebra (i.e., union, set difference, selection, Cartesian product, projection etc.). The motivation for this translation lies in the fact that the relational model is the underlying mathematical model of data for Datalog and there are a number of database optimization techniques applicable for the relational model. Finally optimized relational expressions serve as an input for computing the meaning of recursive Datalog programs.

The core of the IRIS architecture, see Figure 1, is defined as a layered approach consisting of:

- Knowledgebase API;
- Invocation API;
- Storage API.

The knowledgebase API is a top API layer encapsulating central abstractions of the underlying system (e.g., rule, query, atom, tuple, fact, program, knowledge base, context etc.). The purpose of this layer is to define the basic concepts of data model used in IRIS as well as to define the functionality for the knowledge base and program manipulation.

---

[7] http://sourceforge.net/projects/iris-reasoner/

[8] WSML2Reasoner framework: http://tools.deri.org/wsml2reasoner/

[9] IRIS is continuously being developed and the support for non-stratified negation and unsafe rules is envisioned in coming releases.

The invocation API characterizes a particular evaluation strategy (e.g., bottom-up, top-down or mixture of these two strategies) and evaluation methods for a given strategy which are used with respect to a particular logic program.

IRIS implements the following evaluation methods[10]:

- Naive evaluation;
- Semi-naive evaluation;
- Query-subquery (QSQ) evaluation.

The storage layer defines the basic API for accessing data and relation indexing. A central abstraction in this layer is a relation which contains a set of tuples and serves as an argument in each operation of relation algebra. The implementation of IRIS relation is based on Collection and SortedSet Java interfaces where red-black binary search trees are utilized for indexing.

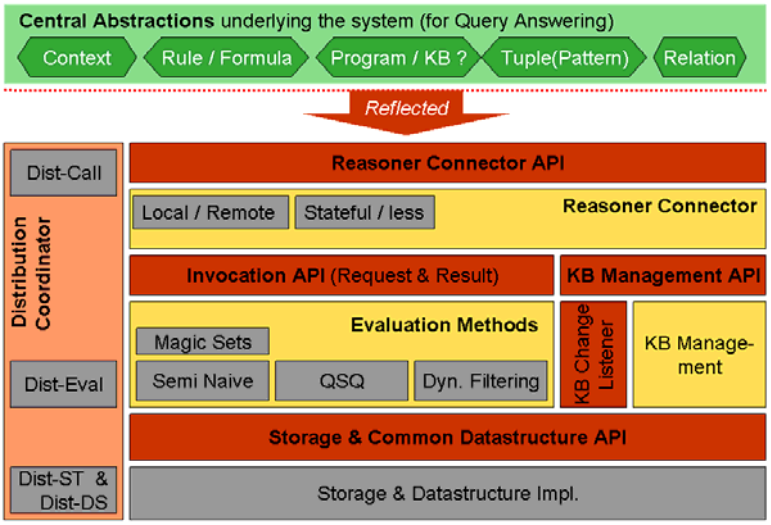

Figure 1: IRIS Architecture

Current inference systems exploit reasoner methods developed rather for small knowledge bases. Such systems either process data in main memory or use a Relational Database Management System (RDBMS) to efficiently access and do relational operations on disk persistent relations. Main memory reasoners cannot handle datasets larger than their memory. On the other side, systems based on RDBMSs feature great performance improvement comparing with main memory systems, but efficient database techniques (e.g., cost-based query planning, caching, buffering) they utilize are suited only for EDB relations and not fully deployable on derived relations.

IRIS is designed to meet requirements for large scale reasoning. Apart from the state-of-the-art deductive methods, the system utilizes database techniques and ex-

[10] More evaluation techniques are under development.

tends them for implicit knowledge in order to effectively process large datasets. We are building an integrated query optimizer. The estimation of the size and evaluation cost of the intentional predicates will be based on the adaptive sampling method [LN90, RR06], while the extensional data will be estimated using a graph-based synopses of data sets similarly as [SP06]. Further on, for large scale reasoning (i.e., during the derivation of large relations which exceeds main memory), run time memory overflow may occur. Therefore in IRIS we are developing novel techniques for a selective pushing of currently processed tuples to disk. Such techniques aim to temporarily lessen the burden of main memory, and hence to make the entire system capable of handling large relations.

The comparison shows that a RDBMS with integrated IRIS inference engine is the only suitable solution to fulfill the requirements of the SBPR.

4. Overall Architecture

In this section, we present the overall architecture of the SBPR. The BPL has been designed in a layered architecture style consisting of

- Semantic Business Process Repository API
- Service Layer
- Persistence Layer

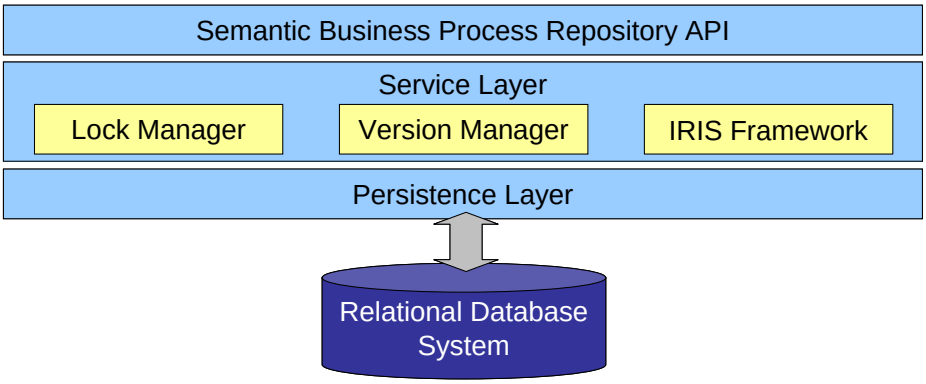

Figure 2: SBPR Architecture

Semantic Business Process Repository API

The Semantic Business Process Repository API provides programmatic access to the SBPR. It includes an API realizing the CRUD pattern, which represents the four basic functions of persistent storage, namely create, retrieve, update and delete. Besides the CRUD API, the SBPR API also provides check-in and check-out operations for long-running process modeling. The Query API rounds off the SBPR API by providing programmatic access to the IRIS Framework for query answering.

Service Layer

The Service Layer implements the SBPR API and processing logic of the SBPR. The Service Layer contains three modules: Lock Manger, Version Manager and the IRIS Framework. The Lock Manager takes charge of requests on locking and unlocking of the process models in the SBPR. A locking request can only be granted when the process model is not yet locked. The Version Manager takes care of the management of the change histories of process models. To record the change history every new process model or changed process model is stored as a new version in the SBPR. IRIS Framework takes the responsibility for the query processing in SBPR.

Persistence Layer

The Persistence Layer manages the data access to the underlying relational database system and provides an abstraction for data access operations. It provides persistent solutions for persistent objects by adopting Object Relational Mapping (ORM) middleware such as Hibernate [HIBER] and Data Access Object (DAO[11]) pattern.

5. Summary and Outlook

In this paper we have presented a Semantic Business Process Repository (SBPR) for storage and management of semantic business process models in SBPM. We have first analyzed the main requirements on the SBPR. After the comparison of different approaches for storage mechanisms, we concluded that an RDBMS with an integrated IRIS inference engine is the most suitable solution, due to the expressiveness of the query language and the reasoning capability..

Currently IRIS is a WSML-Flight reasoner. The system is extensively being developed to support reasoning with WSML-Rule (i.e., support for function symbols, unsafe rules and non-stratified negation). Further on, IRIS will tightly integrate a permanent storage system designed for distributed scalable reasoning. One of our major objectives is the implementation of Rule Interchange Format (RIF)[12] in IRIS. Implementing RIF, IRIS will be capable of handling rules from diverse rule systems and will make WSML rule sets interchangeable with rule sets written in other languages that are also supported by RIF.

Finally, IRIS will implement novel techniques for reasoning with integrating frameworks based on classical first-order logic and non-monotonic logic programming as well as techniques for Description Logics reasoning.

[11] http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

[12] Rule Interchange Format-W3C Working Group: http://www.w3.org/2005/rules/

Acknowledgements

We would like to thank Marin Dimitrov, Graham Hench, Monika Kaczmarek, Dr. Dimka Karastoyanova, Mihail Konstantinov, Tammo van Lessen, Prof. Dr. Frank Leymann, Jörg Nitzsche, Jussi Vanhatalo, Karol Wieloch, Paweł Żebrowski and all other colleagues from the SUPER project for valuable discussions. This work has in part been funded through the European Union's 6th Framework Program, within Information Society Technologies (IST) priority under the SUPER project (FP6-026850, http://www.ip-super.org).

References[13]

[ACG+05] Andrews, Tony; Curbera, Francisco; Dholakia, Hitesh; et al.: Business Process Execution Language for Web Services Version 1.1. 5 May 2003

[AHV95] Abiteboul, Serge; Hull, Richard; Vianu, Victor: Foundations of Databases. Addison-Wesley, 1995

[BD94] Bernstein, Philip A.; Dayal, Umeshwar: An Overview of Repository Technology. In VLDB 1994.

[BKK06] Bruijn, Jos de; Kopecký, Jacek; Krummenacher, Reto: RDF Representation of WSML. 20 December 2006

[BLP+06] Bruijn, Jos de; Lausen, Holger; Polleres, Axel; Fensel, Dieter: The web service modeling language: An overview. In Proceedings of the 3rd European Semantic Web Conference (ESWC2006), number 4011 in Lecture Notes in Computer Science, pages 590–604, Budva, Montenegro, June 2006. Springer-Verlag.

[BPMN06] Business Process Modeling Notation Specification. OMG Final Adopted Specification, February 6, 2006

[GHV+03] Grosof, Benjamin N.; Horrocks, Ian; Volz, Raphael; Decker, Stefan: Description Logic Programs: Combining Logic Programs with Description Logic. In Proc. of WWW 2003, pages 48–57, 2003

[HIBER] Hibernate Reference Documentation. Version: 3.2.0.ga

[HLD+05] Hepp, Martin; Leymann, Frank; Domingue, John; Wahler, Alexander; Fensel, Dieter: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. Proceedings of the IEEE ICEBE 2005, October 18-20, Beijing, China, pp. 535-540

[HR07] Hepp, Martin; Roman, Dumitru: An Ontology Framework for Semantic Business Process Management, Proceedings of Wirtschaftsinformatik 2007, February 28 - March 2, 2007, Karlsruhe (forthcoming).

[KNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)", in: Scheer, A.-W. (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Saarbrücken 1992.

[KOM05] Kiryakov, Atanas; Ognyanov, Damyan; Manov, Dimitar: OWLIM – a Pragmatic Semantic Repository for OWL. In Proc. of Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), WISE 2005, 20 Nov, New York City, USA.

[13] All hyperlinks used in this paper are followed at April 10, 2007.

[LL01] Libkin, Leonid: Expressive Power of SQL. The 8th International Conference on Database Theory. London, United Kingdom, 2001

[LN90] Lipton, Richard and Naughton, Jeffrey. Query size estimation by adaptive sampling (extended abstract). In PODS '90: Proceedings of the ninth ACM SIGACTSIGMOD-SIGART symposium on Principles of database systems, pages 40–46, New York, NY, USA, 1990. ACM Press.

[OWLIM] OWLIM – OWL semantics repository. 2006. http://www.ontotext.com/owlim/

[PT04] Passin, Thomas B.: Explorer's Guide to the Semantic Web. Manning, 2004.

[RDF04] RDF Primer, W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-primer

[RDFS04] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004

[RR06] Ruckhaus, Edna and Ruiz, Eduardo. Query evaluation and optimization in the semantic web. In Proceedings of the ICLP'06 Workshop on Applications of Logic Programming in the Semantic Web and Semantic Web Services (ALPSWS2006), Washington, USA, August 16 2006.

[SF03] Smith, Howard; Fingar, Peter: Business Process Management. The Third Wave. Meghan-Kiffer,US 2003.

[SKS06] Siberschatz, Abraham; Korth, Henry F.; Sudarshan, S.: Database System Concepts. Fifth Edition, McGraw-Hill, 2006.

[SP06] Joshua Spiegel and Neoklis Polyzotis. Graph-based synopses for relational selectivity estimation. In SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pages 205–216, New York, NY, USA, 2006. ACM Press.

[SUPER] The European Integrated Project – Semantics Utilised for Process Management within and between Enterprises. http://www.ip-super.org/

[TH05] ter Horst, Herman J.: Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity. In Proc. of ISWC 2005, Galway, Ireland, November 6-10, 2005. LNCS 3729, pp. 668-684.

[WSML05] Bruijn, Jos de; Lausen, Holger; Krummenacher, Reto; Polleres, Axel; Predoiu, Livia; Kifer, Michael; Fensel, Dieter: The Web Service Modeling Language WSML. 5 October 2005. http://www.w3.org/TR/rdf-schema/