Automating ontology based information integration using service orientation

BOSTJAN GRASIC, VILI PODGORELEC
Faculty of electrical engineering and computer science
University of Maribor
Smetanova ulica 17, 2000 Maribor
SLOVENIA
bostjan.grasic@uni-mb.si, vili.podgorelec@uni-mb.si

Abstract: With the rise of the Internet, globalization and the increasing number of applications used inside organizations, there is an emerging need to integrate information across heterogeneous information systems. Service oriented architecture (SOA) is seen as a general answer to intraorganisational as well as interorganisational integration problems. While service oriented systems have been well studied, there are still some challenges remaining unanswered. One of them is automation of service execution. This paper proposes a method for automated execution of Web Services. Based on Web Service execution automation, the proposed approach is bridging the gap between ontology based integration and service oriented architecture by enabling dynamic and transparent integration of information which is provided by services.

Key-Words: Service execution, Web services, Soa, Information integration, Semantic Web

1 Introduction

Information system engineering nowadays deals greatly with system interoperability issues, information integration being one of the most significant ones. Ontology based integration is one of the possible solutions to the information integration challenge. While it has many advantages [4][9] there are still some gaps. One of the most significant ones, in our opinion, is automation of the data integration part. We propose incorporation of SOA principles to ontology based integration.

Services can be seen as data provision technology, where the data they provide can be integrated at a higher level - the ontology level. The main challenge is how to enable automated and transparent service invocation. This paper proposes a novel approach to ontology based integration, which is based on automated execution of Web Services and automated integration of data they provide in a global data view.

The main advantage of this approach is that it is suitable for any heterogeneous information system, as long as a specific system is able to expose their data in form of services. Data retrieval, conversion to semantic form and integration is taken place automatically when the need for a specific data arises. The described approach as well as automation of the service execution is described in more detail in the rest of the paper.

The organization of the paper is as follows. First section briefly summarizes ontologies and semantic technologies. In the next section, ontology based integration and existing state of the art for integrating semantic and non-semantic data is described. Section 3 introduces Semantic Web Services. In Section 4, the service execution engine, which automates service execution on the need basis, is described. Section 5 describes a use case, where automated service execution would benefit to traditional approaches. In Section 6, performance results are compared to competing approaches, while the last section concludes the paper.

2 Related Work

The field of interapplication integration has a very rich research activity [4][9][10]. One of the factors is the number of different applications used in a single organisation. While this number is rising, there is even greater tendency to share data among applications. Panian defines various levels of integration complexity: from simple data transport, which involves moving data objects between systems, to ubiquitous integration, which allows anytime, anywhere integration through standard means [10]. While ubiquitous integration is a visionary goal, Panian defines different stages in between. The stages from the most primitive, to the most complex integration stage are as follows: data integration – synchronisation of data between systems, application integration – leveraging functionality in applications, process integration –

integration at business process level, collaboration – combining applications, data and human resources in the enterprise [10].

In the recent years, service oriented architecture (SOA) gained a momentum and is seen as a step from application integration to process integration. Several authors are identifying advantages over classic enterprise application integration (EAI) [13][14][25]. Major advantages are better business agility due to improved flexibility and better cost efficiency by empowering reuse and limiting the number of connections between applications.

The approach presented in this paper uses services, as they are defined in the context of SOA, in the role of data providers. In this manner, the approach bridges the gap between ontology based systems and contemporary SOA ecosystems. At the same time, the presented approach enables ontology based integration [4][9] by reusing services from existing SOA systems.

Most significant and novel contribution of the presented work is automation of service execution. The whole process of Web Service execution and data integration is fully automated. The key enabler is semantic service description, which enables declarative execution of services, as well as declarative integration of data returned by services.

There has been some research work done in the area of service execution automation; however none of existing works, in our knowledge, concentrates on interoperability between data represented in semantic networks and web services. Almost all of the research work done in the area of semantic web services is oriented towards discovery and composition of services. There is a working group under Organization for the Advancement of Structured Information Standards (OASIS) [26] called semantic execution environment (SEE). However, the working group limits their activities only to WSMO [8], which is essentially designed for service composition. SEE provides a reference implementation of an execution environment called WSMX for WSMO based services [20].

Martinek et al. has developed a system based on semantic service descriptions for enterprise application integration [12]. Their approach is similar to ours technology wise, however they mostly address data mediation for the purpose of ubiquitous service composition. In this manner, their approach is fundamentally different because they employ semantic technologies on the services level, while we use services on the semantic level in order to bridge semantic applications and SOA infrastructure in a transparent manner. The importance of incorporating contemporary approaches as SOA into knowledge systems is stressed out by Chang and Tseng [11]. Our approach can also be seen as an enabler platform for connecting knowledge system to the SOA infrastructure.

Fundamentally most similar approach to ours has been presented by Langegger et al. They have created a system for virtual data integration that uses SparQL endpoints as data [21]. Similar to our solution, they integrate various data sources on the declarative level. The actual integration is done transparently in run-time when a user makes a standard query. The major difference is that they use Sparql endpoints as data sources, while our approach uses Web Services as data sources.

The presented approach may be used in different independent domains. While this article sketches possible use in the e-tourism domain, papers [18] and [19] describe the use of the presented approach in medical information systems. This approach reduces the size of the semantic network by exposing a part of data in form of services. As identified by Soto et al.[15], semantic web servers may not perform efficient on large datasets. By reducing the size of the semantic network, the performance of the reasoning and retrieval systems may be improved.

3 Ontologies and Semantic Technologies

Ontologies are one of the key technologies in the evolving Semantic Web. There exist several languages as well as several formalisms to capture knowledge and represent it in ontologies. According to the most recent survey that analyzed the use of ontological languages, OWL is being by far the most used one. Over 75 percent of respondents answered they are using OWL [3].

Web ontology language (OWL) [7] is based on description logic and is one of the main building blocks of the Semantic Web technologies (SWT). SWT is a set of technologies, tools and recommendations proposed by World Wide Web consortium (W3C), that follow the vision of semantic web. The vision of semantic web is in evolution from web of documents (as we know the World Wide Web today) into semantic web (SW). In SW, computers will have an awareness of the

meaning of data; hence computer agents will be able to find and process information based on their meaning [2].

SWT provides technologies to achieve this vision. Core technologies that SWT are built upon are Unicode, URI and XML. This foundation enables SWT to be platform and programming language independent. Upon XML is RDF layer. RDF (Resource Description Framework) [27] is a XML-based language for describing resources. On top of RDF is ontology layer. OWL, which is based on RDF, is used as the ontology language. On top of ontology layer is logic layer. This layer enables to define additional rules in rule interchange format (RIF).

4 Ontology based integration

Inter-application interoperability has been long seen as schema mapping and data integration problem. In this manner integration requires (1) mapping systems that define relationships (mappings) among schemes and (2) integration systems that use those mappings to answer queries or translate data across data sources [4].

According to [9] there are three different categories of ontology based integration approaches: single ontology approaches (SOIA), multiple ontology approaches (MOIA), hybrid ontology approaches (HOIA). SOIA use single upper ontology to which all other systems conform. MOIA use multiple interconnected ontologies, each system having its own. HOIA on the other hand is seen as a combination of other two. Approach presented in the paper is a SOIA approach, where the mappings are in form of services which are executed automatically.

It is unrealistic to expect that all the data will be in semantic form, which would simplify the integration process significantly. Rather one can expect a mixture of semantic and non-semantic systems, where the former are in minority. Currently there exist two techniques that are used to transform non-semantic data into RDF. These are:

- Export from non-semantic into semantic form and
- Dynamic relational database mapping.

4.1 Export into semantic form

Most straightforward technique is to export non-semantic data into RDF and then import RDF data into the inference system with the preloaded ontology. Usually custom software programs are coded to achieve this. In case data source is in XML format, then a special mechanism called gleaning resource descriptions from dialects of languages (GRDDL) may be used. GRDDL automates the transformation procedure.

Computer agents or scripts are used to transform data from external data sources (e.g. relational databases, text files, spreadsheets, application specific files, etc.) into RDF documents based on concepts defined in the ontology. The reasoning engine is responsible to integrate the data and reason on it. The knowledge that is captured in such system can be used by querying the reasoning engine and displaying the information to the user.

The described approach is easy to set up, but may be costly to maintain. If the exported data is changing frequently, then data integrity issued should be considered and handled. We have to identify when the imported data changes, then we need export that data and import it again into the knowledge system.

4.2 Relational database mapping

Most critical issue with the first presented approach is identification of data changes and repetition of the export-import procedure. This issue may be overcome by providing external data dynamically. State-of-the-art approach to achieve this is by relational database mapping.

This technique defines mappings between relational schemes and concepts in ontologies. RDF data is provided dynamically by the transformation engine. There are several approaches to access data using this technique. Transformation engines allow different ways to access the data. Most useful data access for ontology based information integration is direct access via SPARQL endpoint. SPARQL is a query language to access semantic data. This way the knowledge system can access and query the data as if it was in RDF. Actually, the transformation engine is querying the database and transforming the data automatically when the SPARQL endpoint is queried.

There already exist stable implementations of such transformation engines e.g. D2R, Virtuoso, Triplify, METAmorphoses, SquirrelRDF. The most widely used implementation D2R enables SPARQL access, RDF dump of the complete relational database, as well as access via a Web browser.

4.3 Service oriented interoperability

Both previously described approaches have their shortcomings. In the first approach, one needs to take care about data integrity between non-semantic

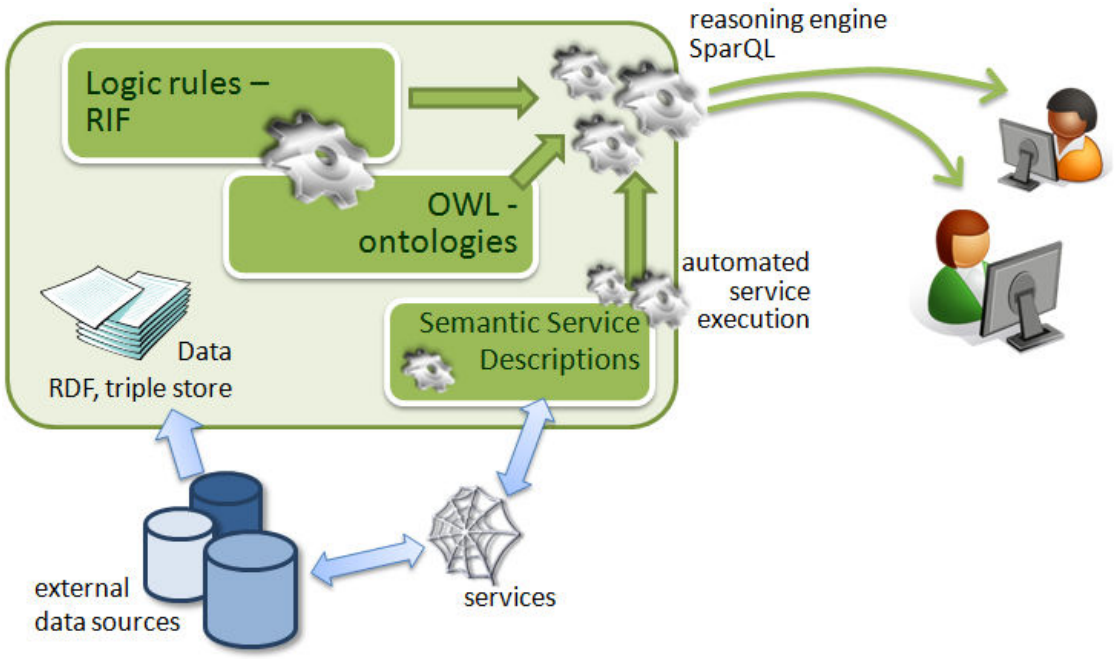

Fig. 1: Otology base integration architecture

data and data in the knowledge system. The second approach solves this issue by providing data on demand, but there is another issue with the second approach. Data is available via a separate, virtual SPARQL endpoint. This means that transformed non-semantic data, which should be used by the system, needs to be integrated manually.

The ideal integration technique would need to solve both issues:

- Provide data dynamically – on demand,
- Automatically integrate data with knowledge system' s data.

We propose a novel, service oriented approach (SOA) that meets both criteria. Instead of mappings between relational schema and ontological concepts, we use Web Services as data providers. They can provide dynamic input from an arbitrary data source, not just from relational databases.

Service oriented architecture addresses shortcomings of the enterprise application integration (EAI) approach. The main shortcoming of EAI and hence also of direct database to database mappings are high maintenance cost. In order to connect n applications, $n * (n - 1)$ connections have to made - between each application pair [25].

Basic idea behind SOA is: applications should expose their data and functionality in form of services, which provide platform and programming language independent communication interfaces. Applications that need other application's data need only to invoke appropriate service. Because each application exposes data and functionality other applications may need, the number of hardcoded and unique connections between applications can be drastically reduced [25].

The proposed ontology based system architecture that is using service oriented approach is shown in Figure 1. External data sources define regular Web Services that are used as data providers for the knowledge system. The proposed framework and also the implemented system do not require development of new Web Services. Existing Web Services can be reused by adding semantic descriptions to their inputs and outputs. Based on these descriptions, Web Services are automatically executed and data that is returned by them is automatically integrated into the knowledge system.

The core component of the proposed technique is the Service Execution Engine (SEE). SEE integrates itself into the reasoning engine and identifies when the data, which is provided by Web Services, is needed. When this need arises, SEE:

- identifies which services need to be executed in order to provide this data,
- prepares input messages Web Services,
- executes Web Services,
- integrates data returned by Web Services into the knowledge system.

The proposed architecture enables fully transparent and dynamic integration of non-semantic data for the ontology based information integration. We have developed the described service execution engine which implementation is described more detail in the rest of the paper.

5 Semantic Web Services

Web Services are seen as the technology of choice for implementing service oriented architecture (SOA) systems. While they provide state of the art data exchange platform for heterogeneous environments (being platform and programming language independent), they lack in automated service discovery and execution aspects.

Current WS descriptions rely only on syntax for defining WS interfaces. To be able to use a service, the consumer has to know what the operations of particular WS actually do and what is the meaning of data they return (XML Schema is not sufficient for this task). WS specifications do not provide means to describe this in a formal way.

These problems led to defining Semantic Web Services (SWS) [1], which is an approach that tries to combine Web Services with Semantic Web concepts. Main idea behind SWS is: if we define the semantics of WS operations and data that is being exchanged with the service in a formal computer readable and processable way (vision of Semantic Web), then we can automatically discover, compose and execute web services.

There are three main approaches to SWS; these are: WSMO [8], OWL-S [6] and SAWSDL [5]. WSMO and OWL-S are mainly concentrating on service discovery and composition. For these reason, they provide fairly complex ontologies to semantically model services. Both approaches enable use of Web Services for data exchange. Main difference between these two concepts is that OWL-S uses SWT for service modeling purposes (RDF, OWL, SAWSDL), while WSMO introduces its own language called WSML [16].

WSMO and OWL-S require a lot of effort to provide semantic service descriptions; SAWSDL, on the other hand does not define the way services are modeled, rather it just provides a mechanisms to semantically annotate existing web service descriptions (WSDL) [17]. How or in which language the consumer defines these concepts is out of scope of the specification. Because of that, SAWSDL is seen as an iterative approach from WS to SWS.

We chose to use SAWSDL in our system, because of following facts: (1) defining WSMO and OWL-S services is a complex task, besides that we do not need discovery and composition capabilities in our system; thus we can use a simpler formalism, (2) WSMO uses its own language, that is not compatible with SWT, (3) research and development effort of OWL-S is fading and a lot of tools are already outdated, (4) existing WS can be easily converted to SAWSDL by just semantically annotating WSDL, (5) SAWSDL is not just compatible with SWT, but it is also compatible with current Web Services, (6) SAWSDL is interoperable with SOA implementations.

As we already mentioned, SAWSDL does not specify how the services are modeled. Because of that, we have developed a lightweight service modeling ontology that is targeted at automated execution of web services. The ontology is called semantic web services execution ontology (SWSEO).

6 Automated integration execution environment

The core component of the proposed approach is the service execution engine. The system as whole is called Semantic Web Services Execution Environment (SWSEE). SWSEE uses SAWSDL, which is a W3C recommendation, for specifying relations between Web Services and concepts defined in domain ontology. The architecture of the system is shown in Figure 2.

The system as whole, acts like a wrapper to the inference engine's SPARQL endpoint. This way SWSEE is able to identify which data is needed by the query and execute Web Services. The whole process of service input retrieval, data conversion, service execution and data integration is transparent to the user or the developer; the user has to provide only the SPARQL query.

Main components of the architecture are shown in Figure 2. SPARQL query processor is responsible for splitting the query into two subqueries: (1) static query and (2) dynamic query. Static query is addressing only data that is permanently in the

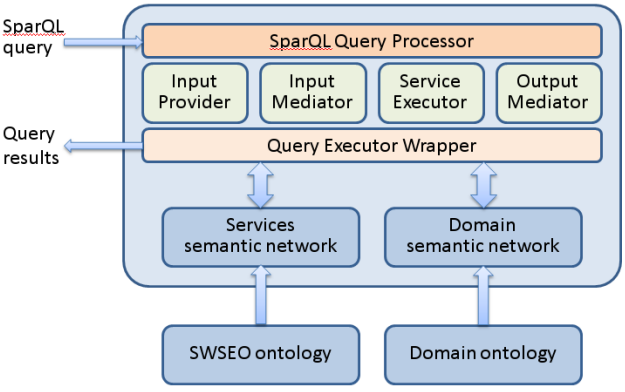

Fig. 2: Component model

knowledge system, i.e. not provided by Web Services. The execution engine executes at first only the static query in order to identify for each instances Web Services need to be executed. Results from the static query are used to generate service input. Dynamic query is basically the same as the input query. It is executed after the data from Web Services has been integrated into the knowledge system. The query processor component has another important task. It is responsible for identifying Web Services that need to be executed in order to execute the query successfully.

Input provider is responsible for preparing input data for Web Services. A single Web Service may be executed many times. E.g. let us suppose we have a Web Service that returns temperature for a specific geographical location. If the result of the query should contain only one geographical location, then temperature Web Service is executed only once. On the other hand, if the query contains many geographical locations, then the Web Service is executed once for each location. Input provider is responsible for identifying such situations, and for preparing the input according to Web Service"s description.

Input and output mediators are responsible for converting data into the right form. Input mediators convert data from semantic form into XML messages according to Web Service' s specification. Output mediators convert XML messages returned by Web Services into RDF messages. Both types of mediators can be defined either as Java classes or XSL transformations.

Service executor is responsible for executing web services. The query executor wrapper provides data from knowledge system' s domain semantic network as well as from semantic network that holds information about Web Services. The main goal of the executor wrapper is to provide the data in an efficient manner. This is achieved by data caching, which reduces the number of query executions.

7 Prototype architecture

Listing 1 shows the algorithm for automated execution of web services. The algorithm is based on the presented system architecture and is implemented in the prototype described in Section 9. The algorithm and the prototype were developed as a proof of concept on the one hand and as a means for measuring the efficiency of the system as a whole on the other hand.

```
define extQuery(staticQuery,
 serviceOutputConcepts, userServiceInput)
define execItem(inputData, outputData,
 invocationDesc)
define execPlan

extQuery = parseSparQLQuery(inputQuery)
staticData = executeQuery(extQuery.staticQuery)

services[] = getServicesWithOutputConcepts(
 extQuery.serviceOutputConcepts)
foreach (service:services) do
begin
  mainConcepts[] =
 staticData.getMainConcepts(service)
  foreach (mainConcept:mainConcepts)
  begin
    execItem = new execItem()
execItem.inputData.add(extQuery.userService
 Input)
    automaticInput = getAutomaticInput(service,
 mainConcept)
    execItem.inputData.add(automaticInput)
execItem.invocationDesc(service.getInvocation
 Desc())
    execPlan.add(exeItem)
  end
end
foreach (execItem:execPlan.getItems())
begin
  execItem.lowerMessages()
  execItem.invokeService()
  dynamicData = execItem.liftMessages()
end
return staticData + dynamicData
```

Listing 1: *Algorithm for automated WS execution.*

Figure 3 shows the sequence diagram of the message flow during the execution of a query, which is intercepted by the sparql query processor. The query processor is denoted as an instance of the SparqlWrapper class in the sequence diagram. Priori the start of the message flow, as shown in Figure 3, the query processor splits the query into two subqueries and creates static input data, as it was described earlier.

The SparqlWrapper object creates an instance of the AggregatedQuery class, which holds all the information regarding a single query. The input parameters to the constructor shown in Fig. 3 are required in order to execute the service successfully. The list of services is constructed automatically based on the dynamic query.

After preparing the aggregated query, the execution plan is prepared. For each service one or many execution items are created. The number of execution items for each service depends on the service type as well as on the input query. After executing services according to the plan, the static query is processed and result is being integrated with services output.

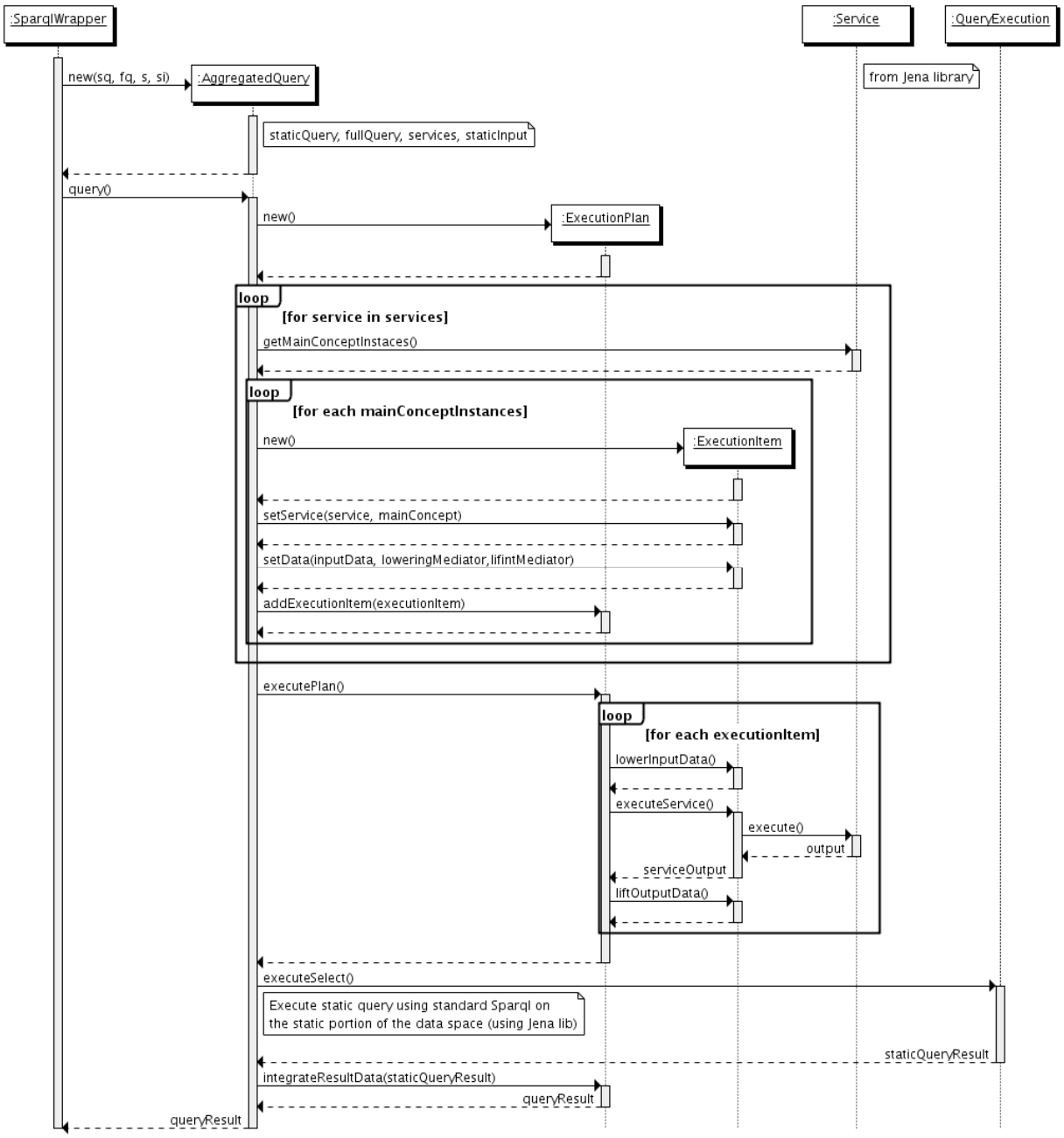

Fig. 3: Query execution sequence diagram

8 Use case

The presented approach is best described on a use case. One of the possible scenarios, where such a system would be of an advantage is in e-tourism. Suppose there is a network of accommodation providers (e.g. resorts, hotels, private apartments). The main objective is to integrate the data about each provider in a dynamic and transparent manner. Data in such system can be static - e.g. address of the accommodation provider, capacity, services offered - or dynamic - available capacities for a specific period, special offers, events.

Static data can be expressed in RDF and provided in a standard Semantic Web manner. On the other hand, dynamic data can be provided by Web Services, which are executed on a need basis - in the same time, their data is being integrated dynamically into the central knowledge base. In order to achieve this, a integration ontology is needed. This ontology can be seen as a common schema for a specific domain or task - in our case providing accommodation information. If accommodation providers expose their static data in RDF according to the domain ontology, then this information can be integrated dynamically by existing approaches. A sample upper ontology for the described use case is shown in Figure 4.

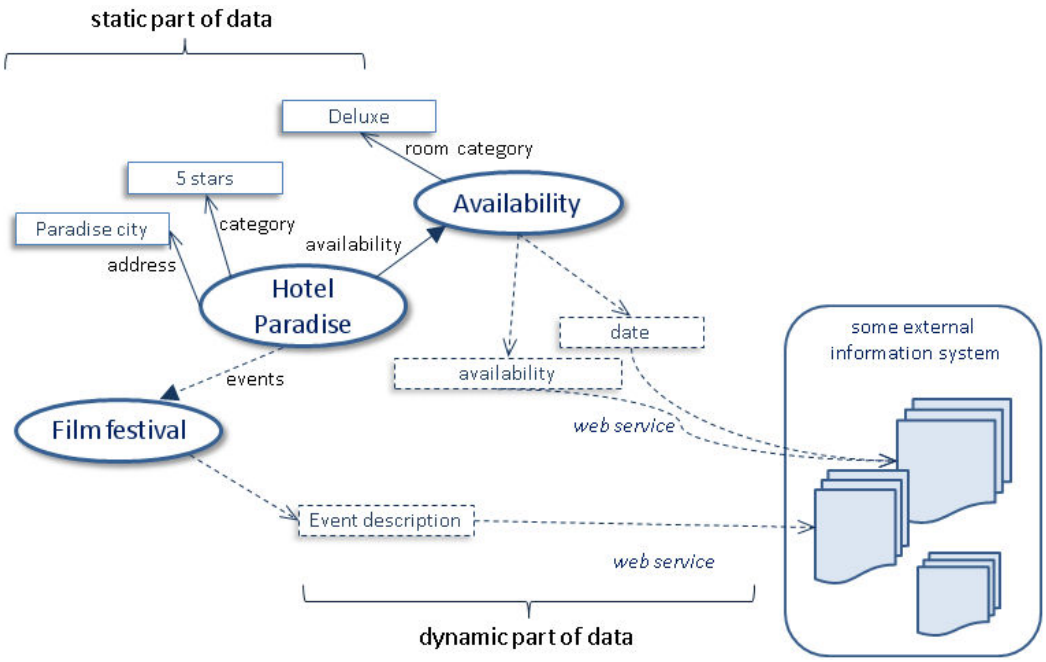

Fig. 4: E-tourism ontology using SWSEE approach

The algorithm for automated service execution is described on the given example. The system is queried using standard Semantic Web query language SPARQL. However, instead of passing the query to the knowledge base, the query is passed to the service execution engine. Based on semantic service description, the execution engine identifies that some of the data, which is being asked for, is being provided by Web Services. The execution engine then identifies services that need to be invoked and prepares their input data. There are two types of services, we shall call them broad and narrow.

For broad services, there exists only one service for a specific kind of data. A example service might be a weather service, which provides weather forecast information. In our case, there is only one service for all locations. However, for the capacity availability information, there exist several services that all provide the same data - namely availability information. The major difference is, that each of those services provides availability information for a specific accommodation provider. A hotel may have its own availability service that is exposed from its internal information system, while a group of private apartments may use shares service, which is provided by the local tourist office. The execution engine identifies services for each and every kind of data.

After the engine identifies which services are about to be executed, it prepares their input data from the knowledge base or from the query itself. For example, period of stay is naturally provided within the query itself, while some other information might be stored in the knowledge base. In the former case, the engine takes information from the query, while in the latter case the execution engine grabs information directly from the knowledge base. After the input data has been prepared, the services are invoked and data, which they return, is integrated into the knowledge base dynamically using XSLT transformations that transform data from XML into RDF.

9 Performance evaluation

A reference implementation of the proposed system was developed using Java programming language and Jena libraries. Plain WSDL based Web Services have been developed using JAX-WS. They were deployed on Sun's Glassfish application server. Generated WSDL documents were annotated with SAWSDL annotations. XSLT mediators have been used for lifting and lowering of messages. Annotated SAWSDL documents were processed using SAWSDL4J library [22].

The system was tested on real data. We used open source database containing music artists and albums data, called Musicbrainz [23]. The test query requested basic information about artists and the list of albums they have released. We have tested performance on a normal RDF memory store (all data has been imported a priori), relational database mapping engine and the proposed SWSEE.

Evaluation was performed on a computer using a 2 GHz Intel Core 2 Duo processor with 2 GB of memory, running Linux Ubuntu operating system and Sun Application Server 9.1. For each use case, two subsequent batch runs were made. In each batch run, the same query was executed for 12 times. The first run was removed from each batch, because of dynamic class loading and one-time class initialization. Table 1 shows performance comparison between the proposed framework, data stored in Jena RDF memory store and a relational database mapping engine. In the case of RDF memory store, only music artist names and albums were imported into the RDF store. In the remaining two systems, the same database system with the fully loaded Musicbrainz database was used. For relational database mapping engine, D2R server [24] was used.

Classic RDF memory store performed best, however it did not contain the entire Musicbrainz database. It does not provide dynamic data provision, or dynamic data integration. The proposed framework performed slightly better than the relational database mapping engine. Since this is a reference implementation, there is still some room for improvements. However it should be noted, that the reference implementation does not yet implement input query stripping and service identification, however noticeable performance slowdowns are not expected by adding these features.

System	Avg (ms)	St. dev
RDF memory store (1st run)	215,64	49,36
RDF memory store (2nd+ run)	1,82	0,87
Relational database mapping	37,82	11,69
SOA integration framework	36,02	12,68

Table1: Performance comparison

10 Conclusion

This paper describes an approach for automated execution of Web Services, which can be used for ontology based integration. Automation of the service execution approach has been described in detail and compared to existing approaches, which could be used for the same goal – ontology based information integration.

The presented approach has been implemented as a prototype which in turn has been evaluated with the aforementioned approaches. It would be expected, that using Web Services as data providers would have a great impact on latency - especially comparing it to the relational database mapping techniques. However, performance tests have shown that the developed prototype is even slightly faster than the well accepted D2R server. Although the prototype did not implement all the functions described in the paper (query processing), it is not expected that implementing remaining functions would have a greater impact on final results.

There are still some open questions for future work. While the prototype may be seen as a feasibility study for the proposed information integration approach, the problem of splitting the query into static and dynamic query has not been addressed fully. The other relevant future work question is how to prepare the execution plan for interdependent services.

References:

[1] Rama Akkiraju. Semantic Web Service - Theory, Tools, and Applications, chapter Semantic Web Services, pages 191–216. Idea Group, 2006.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: Scientific american. Scientific American, May 2001.

[3] J. Cardoso. The semantic web vision: Where are we? Intelligent Systems, IEEE, 22(5):84–88, Sept.-Oct. 2007.

[4] Dejing Dou and Paea LePendu. Ontology-based integration for relational databases. In SAC '06: Proceedings of the 2006 ACM symposium on Applied computing, pages 461–466, New York, NY, USA, 2006. ACM.

[5] Joel Farell and Holger Lausen (Eds.). Semantic annotations for wsdl and xml schema. W3C Reommendation, August 2007.

[6] David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massiomo Paolucci, Katia Sycara, Deborah L. McGuiness, Evren Sirin, and Naveen Srinivasan. Bringing semantics to web services with owl-s. World Wide Web, 10:243-277,207.

[7] Deborah L. McGuiness. OWL Web Ontology Language. W3C Recommendation, February 2004.

[8] Dumitru Roman, Holger Lausen, and Uwe Keller. Wsmo final draft. WSMO Working Draft, http://www.wsmo.org/TR/d2/v1.3/, October 2006.

[9] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information — a survey of existing approaches. In H. Stuckenschmidt, editor, IJCAI–01 Workshop: Ontologies and Information Sharing, pages 108–117, 2001.

[10] Panian, Z., Trends in IT Empowered Enterprise Integration, WSEAS Transactions on Communications, January 2004, pp. 122-127.

[11] Chang, C. C. and Tseng, C. Development and integration of expert systems based on service-oriented architecture. In Proceedings of the 7th WSEAS international Conference on Simulation, Modelling and Optimization (Beijing, China, September 15 - 17, 2007). D. Xu, Ed. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, 2007, pp. 116-121.

[12] Martinek, P., Tothfalussy, B., and Szikora, B. 2008. Implementation of semantic services in enterprise application integration. W. Trans. on Comp. 7, 10 (Oct. 2008), 1658-1668.

[13] Mahmood, Z. Service oriented architecture: potential benefits and challenges. In Proceedings of the 11th WSEAS international Conference on Computers (Agios Nikolaos, Crete Island, Greece, July 26 - 28, 2007). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, 2007, pp. 497-501.

[14] Panian, Z. The SOA ecosystem. In Proceedings of the 9th WSEAS international Conference on Data Networks, Communications, Computers (Trinidad, Tobago, November 05 - 07, 2007). B. Bhatt, B. Tewarie, A. Lazakidou, and K. Siassiakos, Eds. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, 2007, pp. 360-365.

[15] J. Soto, O. Sanjuan, L. Joyanes. Semantic Web Servers: A New Approach to Query on Big Datasets of Metadata, WSEAS Transactions on Computers, November 2006, pp. 2658-2662.

[16] J. d. Bruijn, H. Lausen, A. Polleres, and D. Fensel. The Semantic Web: Research and Applications, chapter The Web Service Modeling Language WSML: An Overview, pages 590–604. Springer Berlin / Heidelberg, 2006.

[17] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services definition language (WSDL). W3C Note, March 2001.

[18] V. Podgorelec, B. Grašič. Adoption of semantic web technologies for developing medical software systems and services. Information modelling and knowledge bases XXI, (Frontiers in artificial intelligence and applications, Vol. 206). Amsterdam [etc.]: IOS Press, cop. 2010, str. 263-274.

[19] V. Podgorelec, B. Grašič, L. Pavlič. Medical diagnostic process optimization through the semantic integration of data resources. Comput. methods programs biomed.. [Print ed.], aug. 2009, vol. 95, iss. 2, str. S55-S67, doi: 10.1016/j.cmpb.2009.02.015.

[20] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - A Semantic Service-Oriented Architecture. In Proc. International Conference on Web Service. 2005.

[21] A. Langegger, W. Wöß, M. Blöchl. A Semantic Web Middleware for Virtual Data Integration on the Web. In proc. European Semantic Web Conference. 2008.

[22] SAWSDL: Semantic Annotations for WSDL at http://lsdis.cs.uga.edu/projects/meteor-s/SAWSDL viewed 30.10.2009.

[23] Musicbrainz, at http://musicbrainz.org/ viewed 30.10.2009.

[24] D2R Server at http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/ viewed 30.10.2009.

[25] T. Erl. SOA Principles of Service Design. Prentince Hall/PearsonPTR, 2005.

[26] OASIS, Organization for the Advancement of Structured Information Standards at http://www.oasis-open.org viewed 30.20.2009.

[27] F. Manola and E. Miller. RDF primer. W3C Recommendation, 2004.