Knowledge representation with SOA

Daniela Gotseva[1] and Ioannis Dimakopoulos[2]

**[1] Computer Systems Department, Technical University of Sofia
Sofia, Bulgaria**

**[2] Computer Systems Department, Technical University of Sofia
Sofia, Bulgaria**

Abstract

This paper addresses the problem of supporting the software development process through the artificial intelligence. The expert systems could advise the Domain Engineer in programming without the detailed experience in programming languages. He will use and integrate, with the help of deductive database and domain knowledge, the previously developed software components to new complex functionalities.
The objective of this document is to provide the knowledge representation about atomic Web Services which will be registered as the facts in the deductive database. The author proposes to use the decision rules in decision tables for representing the service model which consists of semantic specification, interface description, service quality (QoS), non-functional properties. Also the use of Domain Specific Languages (DSL) for modeling Domain Engineer's re-quests to the expert system will be considered within this document. As the illustrative use case for described knowledge representation the author proposes the domain of SOA-based geographic information systems (GIS) which represent a new branch of information and communication technologies.

Keywords: *domain engineering, Services Oriented Architecture, deductive database, expert system, Domain Specific Languages, service model, complex service.*

1. Introduction

The aim of this document is to propose a new approach of soft-ware development supported by the artificial intelligence. The Services Oriented Architecture (SOA), especially the Web Services go towards the need of developing software families through Domain Engineer which has no detailed experience in computer programming, but has strong expert knowledge. This process could be supported by expert systems.

The background of the consideration is the Domain Engineering approach [8] which relies on developing software families from reusable components which are parts of common domain system. In the future, the software can be named service-ware, where all resources are services in a Service Oriented Architecture. The main idea of this approach is that business processes engineer operates on atomic services, not on the software or hardware that implements the service [9].

The method proposed within this paper could be used in large companies enabled on SOA for realizing business processes management (BPM) applications. Web Services are considered as a promising technology for Business-to-Business (B2B) integration. A set of services from different providers can be composed together to provide new complex functionalities.

2. Concept

Fig. 1 presents the overview of the approach considered within this document. Expert system plays the role of decision supporting system. Its task is to provide the proposition of complex service (workflow of atomic Web Services) basing on the Domain Engineer's request explained by means of Domain Specific Language (DSL). The facts in the deductive database are delivered by Soft-ware Developer which implements new functionalities fashioned as the Web Services compliant with enterprise SOA infrastructure. Software Developer registers the atomic service model into facts database and also the service instance in SOA registrars.

The author of this paper proposed in previous work [3] the proof of concept prototype based on the Java framework for intelligent discovery and matchmaking atomic Web Services within integrated workflow called complex service. Thus, the problem of knowledge representation in Services Oriented Architecture will be considered in next sections.

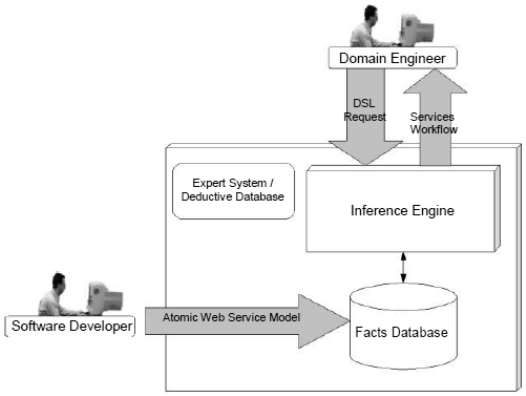

Fig. 1 Overview of the approach.

3. Problem statement and challenges

The solution issue of writing computer program through other computer program is very idealistic challenge, so it seems to be realistic when some assumptions have been fulfilled. The Services Oriented Architecture based on a collection of Web Services that communicate with one another within the distributed systems, which are self-contained and do not depend on the context or state of the other services, allows for discovery of new program functionalities by expert system. The next assumption is that all actors of Fig. 1 should use common domain namespace (domain objects) expressed through domain ontologies (for instance Web Service Modeling Ontology [20]).

The aim of research work described within this document is to provide the sufficient knowledge representation about Web Services which consists of service models, that involve interface de-scription and semantic specification as well as information about service quality (QoS) and non-functional properties.

The properly defined models of atomic Web Services registered as the facts in expert system will enable inferring knowledge about enterprise software resources by Domain Engineer and matchmaking them as the new applications.

4. Related work

The author of [1] describes the semantic service specification, which is the basis for the composition of services to application service processes. Semantic-specified services are a precondition for the development of complex functionality within application service processes. If the user wants to use a service with a desired functionality he sends the semantically specified request and checks which existing services can fulfill this request. The semantic service specification specifies the characteristics of a service. It means, semantic service specification defines what the service does, not how the service doest it. The characteristics of a service contain for example the input parameter, the results, the effects (changing of the world) and the conditions for a successful execution of the service. The first requirement of the semantic service specification is an existing domain ontology, which describes the domain specific concepts and associations and attributes of these concepts. A further requirement for the description of the semantic service specification is a unified description language. The F-Logic language [17] and its extension called Flora-2 [19] have been used. F-Logic is a deductive, object oriented database language which combines the declarative semantics and expressive-ness of deductive database languages with the rich data modeling capabilities supported by the object oriented data model [1].

The authors of this paper propose other approach to explain the service models using Java language expressions. The main objective for this solution is to combine in one programming language: knowledge about services, expert system/rule engine compliant with JSR-94 specification (implementation of the Java Rule En-gine API known as JSR94, which allows for support of multiple rule engines from a single API [16]) as well as J2EE [18] middle-ware and software patterns which is the powerful development platform for Services Oriented Architecture [2]. In the previous paper author proposed the architecture for complex services prototyping and proven the feasibility of this approach on the Java plat-form using the developed prototype [3].

A proper service description answers three questions about a service: what the service does (including its non-functional description), where it is located, and how it should be executed [4]. The Fig. 2 presents the atomic service model proposed by authors of this paper which answers these questions.

Web Services are software applications with public interfaces described in XML. According to the established standards, Web Service interfaces are defined in Web Service Description Language (WSDL) [5]. Published in Universal Description, Discovery and Integration (UDDI) registrars [10] could be discovered and invoked by other software components. These systems interact with Web Services using XML-based message in Simple Object Access Protocol (SOAP).

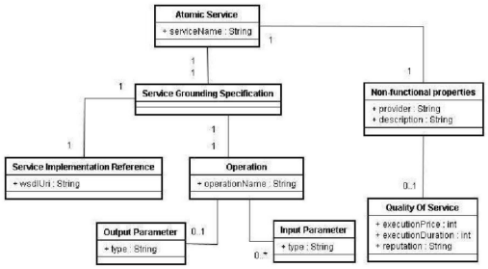

Fig. 2 Atomic Service Model.

Service Grounding Specification (see Fig. 2) refers to the WSDL description. WSDL consists of a hierarchy of objects (proposed within [3] to use domain ontology to define these objects), from the most basic data type, through message, operation, port type, binding and port to service itself [5]. Its wsdlUri attribute is the Unified Resource Identifier (URI) and refers to the service WSDL file. WSDL does not provide methods to describe non-functional service properties.

Quality of Service (QoS) in service oriented platforms is a crucial attribute in assessing proper operation of services. Loosely coupled distributed systems in service discovery, composition and execution have emerged as a new paradigm in building virtual organizations. In order to support rapid and dynamic composition of services it should be possible to locate services that meet user's functional requirements. Moreover, it should be possible to select the best service based on their QoS. It is important to stress the difference between non-functional (NF) and QoS parameters. QoS parameters are a subset of NF parameters. NF parameters may include some information that is not directly computable, for example textual service description, phone numbers to service developers (providers), date of service preparation etc. As a result of that, when using either NF or QoS concepts, one should distinguish that NF relates to a whole set of non-functional parameters, and QoS refers to those NF parameters that may be computationally processed, compared and verified with greater ease [4].

In service arena it is suggested that the term QoS should refer not only to such basic, originating from networking parameters as bandwidth, latency, error rate or availability (the probability that the service is available), reliability (stability of a service function-ality, i.e. ability of a service to perform its functions under stated conditions). Therefore, additional aspects come into consideration, such as speed of operation, robustness, accuracy of operation, de-pendability, capacity (a limit of concurrent requests for guaranteed performance), throughput (the number of requests served in a given time period), response time (the time taken by a service to process its sequence of activities), execution cost (the amount of money for a single service execution). Even parameters such as operating system and storage capacity of the executing system may by considered QoS parameters, as they affect end-to-end op-eration of a service [4] [7].

Currently, most approaches that deal with quality of services address only some generic parameters such as execution price, execution duration, service availability and reliability [6]. These parameters may be defined as follows [4]:

- Execution price – the amount of money that a service requestor has to pay to the service provider for using the Web Service.
- Execution duration (also called latency time) – mea sures the expected delay in seconds between the moment when a request is sent and the moment when the service is rendered.
- Execution duration is a sum of the processing time and the transmission time.
- Reputation (also called Service quality reputation) is a measure of service trustworthiness. It depends mainly on end user's experience of using the service. Different users may have different opinions on the same service.

5. Implementation

All service instances available in particular domain are treated as the knowledge representation system and can be explained as the decision table which contains production rules. Decision tables specify what decisions should be made when some conditions are fulfilled [11]. This document considers the knowledge reasoning problem employing decision tables' formalism

$$K = (U, A) \quad (1)$$

Where K is the knowledge representation system, U is a nonempty, finite set, called universe, and A is a nonempty set of primitive attributes.

The knowledge representation system which distinguishes the condition and decision attributes can be called decision table T:

$$T = (U, A, C, D) \quad (2)$$

Where C and D called condition and decision attributes are two subsets of attributes.

Any implication

$$\Phi \rightarrow \Psi \quad (3)$$

Is considered as the decision rule and Φ, Ψ are called predecessor and successor respectively.

If Eq. (3) is decision rule and P contains all attributes occurring in Φ (condition attributes) and Q contains all attributes occurring in Ψ (decision attributes) then this decision rule can be called PQ-rule.

Let's consider the real decision table (see Table 1), which represents the knowledge system from geographic information systems domain in Services Oriented Architecture and the facts are explained as the PQ-rules. The use case scenario and the services landscape were described within [3].

Table 1: Real Decision Table.

Operation Name	*Input Parameters*	*Output Parameter*	*Provider*	*Execution Price*	*Execution Duration*	*Reputation*	*Service Name*
P1	P2	P3	P4	P5	P6	P7	Q1
getMap	{Coordinates}	Map	TeleAtlas	5$	12ms	high	GisMap
provideMap	{Coordinates}	Map	GISAtlas	0$	24ms	medium	PrintMap
drawPoint	{Coordinates, Map}	PointMarket	GIS Company	0$	2ms	high	DrawPoint
drawSegment	{Coordinates, Coordinates, Map}	SegmentLine	GIS Company	2$	5ms	high	DrawSegment
computeDistance	{Coordinates, Coordinates}	Distance	ITS	0$	1ms	medium	ComputeSegment Distance

The columns P1-P7 represent the condition attributes and column Q1 represents the decision attribute of the PQ-rule. These PQ-rules are stored as the facts in expert system database.

The Eq. (4) formalizes a possible representation of PQ-rule from Table 1 in accordance to the Eq. (3).

P1=getMap and P2={Coordinates} and P3=Map → Q1=GISMap (4)

The authors of this paper prepared the facts database in terms of production rules regarding Eq. 4 and Table 1 as the Java class which is loaded into the Working Memory of expert system (see code listing 1).

Code listing 1: FactsDatabase Class.

```
public class FactsDatabase { WorkingMemory
rulesEngineMemory;

public FactsDatabase(WorkingMemory rulesEngineMemory) {
this.rulesEngineMemory = rulesEngineMemory;
}

public void activateFacts() { AtomicService as;
Collection inputParameters; QoS qos;
// PQ rule
// P-attributes

as = new AtomicService();
as.setOperationName("getMap");
inputParameters = new ArrayList();
inputParameters.add(new
Coordinates().getClass().getName());
as.setInputParameters(inputParameters);
as.setOutputParameter(new Map().getClass().getName());
as.setProvider("TeleAtlas");
qos = new QoS(); qos.setExecutionPrice(5);
qos.setExecutionDuration(12); qos.setReputation("high");
as.setQos(qos); //Q-attributes
as.setServiceName("GisMap");
as.setServiceDescription("Service creates a map
according to provided longitude and latitude.");
rulesEngineMemory.insert(as);
}
```

As the expert system the JBoss DROOLS [12] rule engine based on the RETE algorithm [13] has been used. Drools imple-ments and extends the Rete algorithm which is called ReteOO, what signifying that Drools has an enhanced and optimized im-plementation of the Rete algorithm for Object Oriented systems [14].

The Domain Engineer models the request to the deductive database as the production rules presented in Eq. (3) manner, also to infer conclusions which results in actions ***" When <conditions> then <actions>"***. The advantage of using rules engine is the declarative programming. Rules are much easier to read than source code. Also the ability of creation of executable domain knowledge repository plays the important role. Domain experts are often a wealth of knowledge about business rules and processes. They typically are non-technical, but can be very logical. Rules can allow them to express the logic in their own terms [12].

The production rule example (code listing 2) shows the strength of proposed approach. The Domain Engineer models the request to the deductive database as the one rule instead of a lot of source code lines and nested loops in structural programming languages or SQL statements. But, the production rules modeling could be much easier through usage of Domain Specific Language (DSL). It is the way of extending the rule to problem domain. Simple DSL can be implemented by lexical processing. In addition, DSL can be used to create front-ends to existing systems or

to express complicated data structures. A DSL is a programming language tailored especially to an application domain: rather than being for a general purpose, it captures precisely the domain's semantics [15]. DSL can act as "patterns" of conditions or actions that are used in rules, only with parameters changing each time [12]. Rules expressed in Domain Specific Language have human-readable form and match the expression used by domain experts [15].

Code listing 2: Production Rule Example.

```
rule "serviceProposition1" when
#conditions
as : AtomicService( outputParameter == "soa-
rules.ontology.Map", qos.executionDuration < 20 , ser-
viceName : serviceName, serviceDescription : serviceDe-
scription )
then #actions
System.out.println( "Proposed service1: " + serviceName
+ " - " + serviceDescription);
End
```

Code listing 3 shows how the rule can be transformed to "patterns" of DSL.

Code listing 3: DSL patterns.

```
 [conditions]

DSL Language expression:
There is an Atomic Service where Rule mapping:
AtomicService(serviceName : serviceName, ser-
viceDescription : serviceDescription)
DSL Language expression:
- output parameter equals "{value}" Rule mapping:
outputParameter == "{value}"
DSL Language expression:
- executionDuration is less than "{value}" msec Rule
mapping:
qos.executionDuration < "{value}"

[actions]

DSL Language expression:
Print service name and service description Rule mapping:
System.out.println( "Proposed service1: " + serviceName
+ " - " + serviceDescription);
```

The usage of "patterns" of Domain Specific Language allows the Domain Engineer to model the request to the expert system and find the desired Web Service in friendly manner as shown on code listing 4.

Code listing 4: Usage of DSL patterns.

```
rule "serviceProposition1"

when

        #conditions
        There is an Atomic Service where
        - output parameter equals "soa-
        rules.ontology.Map"
        - executionDuration is less than "20" msec

then

        #actions
        Print service name and service description

End
```

6. Conclusion

The presented approach allows supporting the Domain Engineer in developing applications from business processes management area. The Domain Engineer has no detailed experience in computer programming, but has strong expert knowledge. He can model the requests to the deductive database as the production rules in human-readable format with usage of Domain Specific Languages instead of several lines and nested loops of Java or SQL code The author discussed within this paper the knowledge representation in SOA explained as the decision tables with atomic service models which involve semantic specification, interface description (WSDL), non-functional properties and quality of services (QoS).

The further research will be focused on refinement of reasoning process with usage of other techniques of the artificial intelligence, development of domain specific languages for GIS domain, storage of the facts before loading to production memory (the traditional solution as the text files is not enough convenient to hold on objects) as well as discovery and matchmaking workflows of complex services.

References

[1] Donath Steffi, Automatic Creation of Service Specifications, 6th Annual International Conference on Object-Oriented and Internet-Based Tech-nologies, Concepts, and Applications for Networked World, Net.ObjectDays Proceedings, pp.79-89, September 19-22, 2005

[2] Hansen Mark, SOA Using Java Web Services, Person Education Inc., Prentice Hall, 2007

[3] Grobelny Piotr, Rapid Prototyping of Complex Services in SOA Architecture, IX International PhD Workshop OWD'2007, Conference Archives PTETiS, vol. 23(1), pp.71-76, 2007

[4] Kowalkiewicz Marek, Current challenges in non-functional service description – state of the art and discussion on rese arch results, Net.ObjectDays Proceedings, Spetember 19-22, 2005 pp.91-96

[5] Christensen, E., F.Curbera, et al. Web Services Description Language (WSDL) 1.1, World Wide Web Consortium (W3C), 2001

[6] Zeng, L., B. Benatallah, et al., Quality driven Web Services Composition. Proceedings of the 12th international conference on World Wide Web (WWW) Budapest, Hungary, ACM Press 2003

[7] Kokash Natallia, D'Andrea, Vinzenzo, Evaluating Quality of Web Ser-vices: A Risk-Driven Approach, Business Information Systems, Witold Abramowicz (Ed.) 10th International Conference BIS 2007 proceedings, LNCS 4439, pp.180-194, Springer, 2007

[8] Czarnecki Krzysztof, Eisenecker Ulrich: Generative Programming – Methods, Tools and Applications, Addison Wesley, Boston, MA, 2000

[9] Ekelhart Andreas et al.: Security Issues for the Use of Semantic Web in E-Commerce, Business Information Systems, Witold Abramowicz (Ed.) 10th International Conference BIS 2007 proceedings, LNCS 4439 pp.1-13, Springer, 2007
[10] UDDI Specifications, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm, accessed January 2007
[11] Pawlak Zdzisław: ROUGH SETS Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, 1991
[12] Proctor Mark et al.: Drools Documentation, http://downloads.jboss.com/drools/docs/4.0.4.17825.GA/html_single/ in-dex.html, accessed January 2008
[13] ForgyC., RETE: A Fast Algorithm for the Many Pattern Many Object Pattern Match Problem, Artificial Intelligence, 19(1), pp.17-37 Sept. 1982
[14] Doorenbos Robert B., Production Matching for Large Learning Systems (Rete/UL), PhD thesis, Carnegie Mellon University, January 31, 1995
[15] Spinellis Diomidis, Notable design patterns for domain-specific languages, The Journal of Systems and Software 56 (2001) pp. 91-99, El-sevier 2001
[16] Toussaint Alex, Java Rule Engine API™ JSR-94, Java Community Proc-ess, http://jcp.org/en/jsr/detail?id=94, BEA Systems, September 2003
[17] Kifer Michael et al.: Logical Foundations of Object-Oriented and Frame-Based Languages, Journal of the Association for Computing Machinery, May 1995
[18] Sun Microsystems, Simplified Guide to the Java 2 Enterprise Edition, http://java.sun.com/j2ee/reference/whitepapers/j2ee_guide.pdf, Accessed January 2008
[19] Yang Guizhen et al.: FLORA-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web, Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE), Catania, Sicily, Italy, November 2003
[20] Dumitru Roman et al.: Web Service Modeling Ontology, Applied Ontology, 1(1), pp. 77-106, 2005

Daniela Gotseva is associate professor, PhD and Vice Dean of Faculty of Computer Systems and Control, Technical University of Sofia, from 2008 with primary research interest of programming languages, system programming, and fuzzy logics. She is a member of the IEEE and the IEEE Computer Society.

Ioannis Dimakopoulos is PhD student at Faculty of Computer Systems and Control, Technical University of Sofia, from 2010, with primary research interest of system programming and service oriented architecture (SOA).