ISBN 978-83-60810-14-9
ISSN 1896-7094

The Expert System Approach in Development of Loosely Coupled Software with Use of Domain Specific Language

Piotr Grobelny
University of Zielona Gora,
Faculty of Electrical Engineering, Computer Science and Telecommunication,
Podgorna 50, 65-246 Zielona Gora, Poland
Email: P.Grobelny@weit.uz.zgora.pl

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 Service Oriented Architecture (SOA) and loosely coupled software allow to fulfill these requirements. The objective of this document is to provide the knowledge representation of atomic Web Services which will be registered as the facts in the deductive database as well as the inferring techniques. Also, the use of Domain Specific Language (DSL) for modeling domain engineer's requests to the expert system will be considered within this document.**

I. Introduction

THE aim of this document is to propose a new approach of software development supported by the artificial intelligence. The loosely coupled software, 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 [7] 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 (SOA). 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 [8].

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 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.

A. Concept

The 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 software developer who 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 (see concept overview in [1]).

The author of this paper proposed in previous work [2] 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 as well as usage of DSL will be considered in next sections.

B. Problem Statement and Challenges

The 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 loosely coupled software 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 described in concept should use common domain name space (domain objects) expressed through domain ontologies (for instance Web Service Modeling Ontology [9]).

The aim of research work described within this document is to provide the sufficient knowledge representation system which consists of rules, facts and Domain Specific Language. The properly defined service models, which involve interface description, semantic specification as well as information about service quality (QoS) and non-functional properties, registered as the facts in expert system will enable inferring knowledge about enterprise software resources by domain engineer and matchmaking them as new applications.

II. Related work

The author of [10] describes the semantic service specification, which is the basis for the composition of services to application service processes. Semantic-specified services are the condition for the development of complex functionality within application service processes. The first requirement of the semantic service specification is an existing domain ontology, which describes the domain specific concepts as well as 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 [11] and its extension called Flora-2 [12] have been used. F-Logic is a deductive, object oriented database language which combines the declarative semantics and expressiveness of deductive database languages with the rich data modeling capabilities supported by the object oriented data model [10].

The author of this paper proposes 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 Engine API, which allows for support of multiple rule engines [13]) as well as J2EE middleware which is the powerful development platform for Services Oriented Architecture [15]. In the previous paper author proposed the architecture for complex services prototyping and proven the feasibility of this approach on the Java platform using the developed prototype [2].

Domain Specific Language is the way of extending the rule to problem domain. 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 [22]. DSL can act as "patterns" of conditions or actions that are used in rules, only with parameters changing each time [19]. Rules expressed in Domain Specific Language have human-readable form and match the expression used by domain experts [22]. The domain engineer models the request to the deductive database as one rule instead of a lot of source code lines and nested loops in structural programming languages or SQL statements.

III. Implementation

Web Services are software applications with public interfaces described in XML. 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 [16]. The Fig. 1 presents the simplified atomic service model proposed by author of this paper which answers these questions.

All model elements has been described in details in previous author's paper [1] therefore only some Quality of Service (QoS) generic parameters such as execution price and execution duration will be considered to illustrate the problem of making decisions in fuzzy surroundings as well as usage of Domain Specific Language.

This paper presents the object-oriented approach to usage rule- and model based expert system [23][6] in development of loosely coupled software on Java platform. The author delivers the solution for creating the object-oriented knowledge databases (deductive databases) which consist of atomic service models stored as the facts as well as for inferring the knowledge through rules.

A. Atomic Service Models as Facts in Knowledge Database

All pieces of information stored in deductive database establish the knowledge representation system. In previous work [1][2] the author proposed the decision tables formalisms to explain the facts as the production rules. Decision tables specify what decisions should be made when some conditions are fulfilled [4].

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

$$K=(U,A,C,D) \qquad (1)$$

Where U is a nonempty, finite set called universe, A is a nonempty set of primitive attributes and C, D $\subset$ A are two subsets of attributes called condition and decision attributes. Any implication

$$\phi \rightarrow \psi \qquad (2)$$

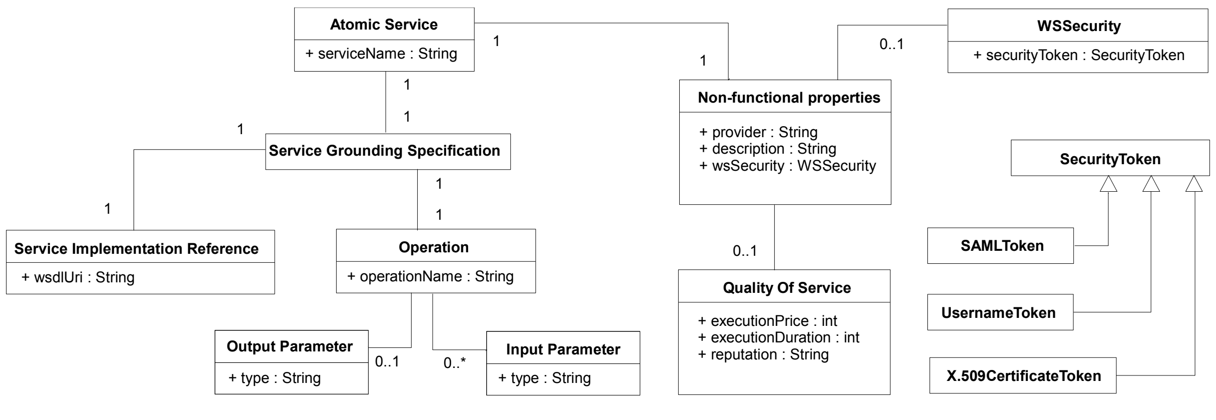

Fig 1. The simplified model of atomic service

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

If $\Phi \rightarrow \Psi$ 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 I) which represents the knowledge system from geographic information systems (GIS) 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 author's previous paper [2].

The columns P1-P6 represent the condition attributes and column Q1 represents the decision attribute of the PQ-rule. These PQ-rules are stored as facts in expert system database. The formula (3) formalizes a possible representation of PQ-rule from Table I in accordance to the formula (2).

$$P1 = getMap \wedge P2 = \{Coordinates\} \wedge P3 = Map \rightarrow Q1 = GisMap \quad (3)$$

The author of this paper prepared the facts database in terms of PQ-rules regarding formula (3) and Table I as the Java class which is loaded into the working memory of expert system. The previous document [1] presents the implementation of the selected fact which expresses the atomic service model with compliance to Jboss Drools [19] object-oriented expert system.

B. Making Decisions in Fuzzy Surroundings

Respectively to formula (3) the facts regarding the membership functions could be explained as the PQ-rules as shown in Table II. Let's consider the example membership functions of fuzzy sets "low execution duration" and "average execution price". These functions should be provided by experts and can be explained in object-oriented manner as suggested in [3].

In the paper [1] has been presented the Java object with implemented membership function for QoS parameter "execution duration" and DSL pattern "low".

The fuzzy sets theory [5][14] allows for making decisions in fuzzy surroundings. The fuzzy set A in certain nonempty space X is defined as collection of couples as shown in (4).

$$A = \{(x, \mu_A(x)); x \in X\} \quad (4)$$

where

$$\mu_A : X \rightarrow [0,1] \quad (5)$$

is the membership function of fuzzy set A.

Formula (6) represents the **symbolic** presentation of fuzzy set A for the finite number of elements X = {x_1, ... , x_n}, $A \subseteq X$

$$A = \frac{(\mu_A(x_1))}{x_1} + \frac{(\mu_A(x_2))}{x_2} + \ldots + \frac{(\mu_A(x_n))}{x_n} = \coprod_{i=1}^{n} \left(\frac{(\mu_A(x_i))}{x_i}\right) \quad (6)$$

where

$$\frac{(\mu_A(x_i))}{x_i}, i = 1, \ldots, n \quad (7)$$

denotes the pair (8) and symbol "+" has the symbolic character, as well.

$$(x_i, \mu_A(x_i)), i = 1, \ldots, n \quad (8)$$

The fuzzy surroundings consists of fuzzy constraints C and fuzzy decision D. Let's consider the set of options X_{op}={x}. The fuzzy constraint is defined as fuzzy set C specified on X_{op} and described through membership function μ_c: $X_{op} \rightarrow [0,1]$.

Let's consider the *n* fuzzy constraints C_1 , … C_n where *n* >1. The fuzzy decision D is determined though formula (9).

$$D = C_1 \wedge C_2 \wedge \ldots \wedge C_n \quad (9)$$

taking into account (10)

$$\mu_D(x) = T\{\mu_{C_1}(x), \ldots, \mu_{C_n}(x)\}, x \in X_{op} \quad (10)$$

where T denotes the t-norm MIN operator as shown in (11)

$$\mu_D(x) = MIN(\mu_{C_1}(x), \ldots, \mu_{C_n}(x)) \quad (11)$$

The maximal decision is the option $\dot{x} \in X_{op}$ selected in the manner described in formula (12)

$$\mu_D(\dot{x}) = max(\mu_D(x)), x \in X_{op} \quad (12)$$

The author took under consideration a situation, when the inference engine proposed three services from Table I regarding to defined constraints C_1 - "low execution duration" and C_2 - "average execution price". These services establish the options collection X_{op}={"MapProvider", "PrintMap", "GisMap"} which are denoted respectively x_1, x_2, x_3. The task of the reasoning process is to take the maximal decision

TABLE I.
THE EXAMPLE OF DECISION TABLE WITH SELECTED ATOMIC SERVICE MODEL ATTRIBUTES

Operation	Input Parameters	Output Parameter	Provider	Execution Price	Execution Duration	Service Name
P1	*P2*	*P3*	*P4*	*P5*	*P6*	*Q1*
getMap	{Coordinates}	Map	TeleAtlas	7$	24ms	GisMap
printMap	{Coordinates}	Map	GIS Atlas	10$	12ms	PrintMap
provideMap	{Location}	Map	GIS Maps	110$	49ms	MapProvider
drawSegment	{Coordinates, Coordinates, Map}	Segment	GIS Company	0$	5ms	DrawSegment
computeDistance	{Coordinates, Coordinates}	Distance	ITS	0$	1ms	ComputeSegmentDistance

TABLE II.
THE EXAMPLE OF DECISION TABLE WITH MEMBERSHIP FUNCTIONS

QoS	DSL Pattern	Membership Function Object
P1	*P2*	*Q1*
execution duration	low	LowExcutionDurationMembershipFunc
execution price	average	AverageExcutionPriceMembershipFunc

$\dot{x}$ as the best proposition regarding the constraints C_1 and C_2.

The formulas (13)(14) specify the fuzzy constraints C_1 and C_2 using symbolic notation from (6) which points out the computed certainty factor (CF) for each option.

$$C_1=\frac{0{,}024}{x_1}+\frac{0{,}95}{x_2}+\frac{0{,}649}{x_3} \quad (13)$$

$$C_2=\frac{0{,}799}{x_1}+\frac{1{,}0}{x_2}+\frac{0{,}4}{x_3} \quad (14)$$

Regarding the formulas (9) and (10) the fuzzy decision D could be denoted as shown in formula (15).

$$D=\frac{0{,}024}{x_1}\wedge\frac{0{,}95}{x_2}\wedge\frac{0{,}4}{x_3} \quad (15)$$

The maximal decision $\dot{x}$ assigned basing on formula (12) points the option x_2—“PrintMap” as the best proposition of atomic Web Service in context of defined constraints.

C. The Use of Domain Specific Language

The domain engineer models the request to the deductive database as the production rules presented in formula (2) manner. These rules allow the experts to express the logic in their own terms [19] of Domain Specific Language (DSL):

```
rule "Service options"
when
    There is a service where
        - output parameter equals "Map"
    There is a service with "low" execution dura-
    tion
then
    Create service options set
end

rule "Make decision in fuzzy surroundings"
when
    There is maximal decision for service options
    set
then
    Propose the best service
end
```

The DSL expressions should be transformed to the notation accepted by shell of expert system:

```
[condition]
There is a service where=
$as: AtomicService ($qos : qos, $serviceName :
serviceName)
[condition]
- output parameter equals "{value}"=
outputParameter == "{value}"

[condition]
There is a service with "{value}" execution dura-
tion=
FuzzyRules ($patternED : pattern == "{value}",
$featureED : feature == "execution duration",
$membershipFuncED : membershipFunc,
eval($membershipFuncED.value($qos.executionDura-
tion) > 0));

[consequence]
Create service options set=
ArrayList al = new ArrayList();
al.add(new Double($membershipFuncED.value($qos.ex-
ecutionDuration)));
al.add(new Double($membershipFuncEP.value($qos.ex-
ecutionPrice)));
insert (new Option ($as, al));
```

The result of `Service options` reasoning rule is the option collection X_{op} with values of the certainty factor (CF):

```
– Option: MapProvider
  low execution duration CF: 0.024,
  average execution price CF: 0.799

– Option: PrintMap
  low execution duration CF: 0.95,
  average execution price CF: 1.0

– Option: GisMap
  low execution duration CF: 0.649,
  average execution price CF: 0.4
```

The outcome of the `Make decision in fuzzy surroundings` rule is the best service proposition (`Proposed service: PrintMap, max membership level: 0.95`) in context of defined constraints established basing on the maximal decision formula as shown in (15).

As the expert system the JBoss Drools [9] rule engine based on the Rete algorithm [20] has been used. Drools implements and extends the Rete algorithm which is called ReteOO, what signifies that Drools has an enhanced and optimized implementation of the Rete algorithm for object-oriented systems [21].

IV. CONCLUSION

The presented approach allows to support 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 programming language code. The author discussed within this paper the knowledge representation in SOA and making decisions in fuzzy surroundings with the help of DSL patterns.

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 as well as discovery and matchmaking workflow of complex services.

The use of expert system seems to be a promising way of programing loosely coupled software systems by the domain experts. Also the Java platform strongly focused on Web Services as well as available object-oriented deductive databases allow for application of proposed solutions in business processes management.

References

[1] P. Grobelny, “Knowledge representation in services oriented architecture”, in *Przeglad Telekomunikacyjny,* 6/2008, SIGMA NOT, pp. 793-796.

[2] P. Grobelny, “Rapid prototyping of complex services in SOA architecture” in *Conference Archives PTETiS*, vol. 23(1), Warszawa, 2007, pp. 71-76.

[3] D. H. Besset, *Object-Oriented Implementation of Numerical Methods, An Introduction with Java and Smalltalk*, Morgan Kaufamann Publishers, Academic Press, 2001.

[4] Z. Pawlak, *ROUGH SETS Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, 1991

[5] L. Rutkowski, *Metody i techniki sztucznej inteligencji*, Wydawnictwo Naukowe PWN, 2006

[6] A. Niederliński, *Regułowo-modelowe systemy ekspertowe rmse*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, 2006

[7] K. Czarnecki, U. Eisenecker, *Generative Programming – Methods, Tools and Applications*, Boston, MA: Addison Wesley, 2000

[8] A. Ekelhart, S. Fenz, A Min Tjoa, E. Weippl, “Security issues for the use of semantic web in e-commerce”, in *Business Information Systems, 10th International Conference BIS 2007 proceedings*, W. Abramowicz, Ed., Springer, 2007, pp.1-13.

[9] R. Dumitru, U. Keller, H. Lausen, J. De Bruijn, L. Ruben, M. Stollberg, A. Polleres, C. Feier, C. Bussler, D. Fensel, “Web service modeling ontology” in *Applied Ontology*, 1(1), IOS Press, 2005, pp.77-106.

[10] S. Donath, “Automatic creation of service specifications” in *Net.ObjectDays Proceedings,* 6th Annual International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for Networked World, 2005, pp.79-89.

[11] M. Kifer, G. Lausen, Wu J, “Logical foundations of object-oriented and frame-based languages” in *Journal of the Association for Computing Machinery,* 1995.

[12] G. Yang, M. Kifer, C. Zhao “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), Italy, 2003.

[13] A. Toussaint, *Java Rule Engine API™ JSR-94*, Java Community Process, BEA Systems, 2003.

[14] J. Kacprzyk, *Wieloetapowe sterowanie rozmyte*, Warszawa: WNT, 2001.

[15] M. Hansen, *SOA Using Java Web Services,* Person Education Inc., Prentice Hall, 2007.

[16] M. Kowalkiewicz, “Current challenges in non-functional service description – state of the art and discussion on research results”, *Net.ObjectDays Proceedings,* 2005, pp. 91-96.

[17] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web Services Description Language (WSDL) 1.1,* World Wide Web Consortium W3C, 2001.

[18] *UDDI Specifications,* http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm , 2007.

[19] M. Proctor, M. Neale, M. Frandsen, S. Griffith, E. Tirelli, F. Meyer, K. Verlaenen, *Drools Documentation*, http://jboss.com, 2008.

[20] C. Forgy, “RETE: A fast algorithm for the many pattern many object pattern match problem” in *Artificial Intelligence,* 19(1), 1982, pp.17-37.

[21] R. Doorenbos, “Production matching for large learning systems (Rete/UL)”, Ph.D. thesis, Carnegie Mellon University, 1995.

[22] D. Spinellis, “Notable design patterns for domain-specific languages” in *The Journal of Systems and Software,* 56, Elsevier, 2001, pp. 91-99.

[23] A. Niederliński, “An expert system shell for uncertain rule- and model based reasoning” in *Methods of Artificial Intelligence,* Gliwice, 2001.