2010 Sixth International Conference on Autonomic and Autonomous Systems

A Rule-based Web Service Composition Approach

Yujie Yao
School of Software
Shanghai Jiao Tong University
Shanghai, China
yaoyujie@sjtu.edu.cn

Haopeng Chen
School of Software
Shanghai Jiao Tong University
Shanghai, China
chen-hp@sjtu.edu.cn

***Abstract*—Web services are rapidly changing the landscape of software engineering. One of the most important technologies introduced by them is Web service composition (WSC). This paper proposes a rule-based service composition framework that supports user requirements and predefined business rules very well. The selection engine uses a multiobjective non-dominated sorting genetic algorithm (NSGA-II) which generates a set of Pareto optimal solutions that provides service consumers wider choices and thus helps increasing the decision support ability. Several experiments have been performed to simulate the selection algorithm and results show that it is effective, efficient and provides a new rule-based mechanism for service composition.**

Keywords-Web service composition; business rule; NSGA-II; Pareto optimal solution

I. INTRODUCTION

In the last few years, the Software as a Service (SaaS) concept has been passed into common usage as a means to license an application to customers for use as a service on demand over the Internet. It is typically thought of as a low-cost way for businesses to obtain rights to use software as needed versus licensing all devices with all applications. Implementing SaaS is typically achieved by leveraging service-oriented architecture (SOA) design principles such as loose coupling and self-describing service interfaces.

As a technology for implementing SOA-based applications, Web services facilitate to achieve this vision by providing specifications, such as WSDL [1] as service description language and SOAP [2] as transport protocol. Practically, we can create "composited services" by composing several existing Web services in a logical order that satisfies user requirements, thus significantly increasing the reusability of existing Web services.

In working with Web service compositions, we realize that a key factor is to model business rules which can help to understand user requirements. The business rules should also be managed automatically by a rule-based system, which is then queried by the business logic to evaluate them.

Our previous study [3] focuses on modeling the QoS-aware service composition problem as a multiobjective optimization problem. And, we also design a corresponding service composition algorithm which based on NSGA-II. In this paper, our trend is to extend the service composition algorithm to support user requirements and pre-defined rules that enhance the composition. We also design a composition framework which allows composing automatically.

This paper is organized as follows. Section 2 presents the related work. Section 3 introduces the business rule. Section 4 describes the composition framework. Some implementation aspects are highlighted in Section 5. Experimental results are shown in Section 6. Section 7 concludes this work.

II. RELATED WORK

Developers used to build a composition by manually specifying the workflow to define the composition logic and selecting available services to suit them. However, this approach poses a big problem, inflexibility. As we know, QoS attributes keep changing all the time. It's probable that services need to be dynamically selected based on the real-time QoS attributes. A static approach cannot handle the situation.

Therefore, dynamic service composition is proposed to fix it. SWORD [4], eFlow [5], and StarWSCoP [6] are typical examples of dynamic service composition systems. They provide ways to support semantic services as they require the post/preconditions and component interfaces that can be defined independently using different annotations.

Other work on Web service composition focused on developing languages and standards to specify and configure the composition process. WS-BPEL [7] provides a language for the specification of Executable and Abstract business processes while OWL-S [8] is an ontology for describing Semantic Web Services. All of them aim to facilitate the automation of Web service discovery, execution, composition and interoperation.

Gerardo et al. [9] proposed a GA-based approach for QoS-aware service composition. They model the problem by means of a fitness function and adopt a static and dynamic penalty function to deal with the constraints. The weaknesses are the same for approaches using fitness function technology. It's unreasonable to combine those different QoSs into one because their measurement units are not the same and the weight factors' assignment is subjective and unstable.

The related work in Semantic Web's area is quite diverse.

The Web Service Modeling Ontology (WSMO) [10] which inherits from the Web Service Modeling Framework (WSMF) [11] is one of the major initiatives in Semantic

978-0-7695-3970-6/10 $26.00 © 2010 IEEE
DOI 10.1109/ICAS.2010.29

IEEE computer society

Web services area. The aim of WSMO is to define an overall framework that covers relevant aspects for Semantic Web Services such as discovery, selection, composition, mediation, execution and monitoring.

The Java Community Process (JCP) released the final version of the Java Rule Engine API (JSR 94) [12] in August 2004. It defines a Java runtime API by accessing a rule engine from a Java Platform. The Java Rule Engine API has already been supported by a couple of rule engines including Drools [13], ILOG Jrules [14], and JESS [15]. Another initiative, focusing on a standard representation of business rules and being a way to standardize a common rule language, is RuleML [16], started in August 2000 and is currently the most promising initiative for representing rule markup for the Semantic Web.

Charfi et al. [17] present a hybrid approach for realizing the integration of business rules (modeled as aspects) using AO4BPEL and integrate a rule-based system with an orchestration engine. Yang et al. [18] define the "Service Composition Lifecycle" that is a methodology for flexible and dynamic construction of application services. Based on that, Orriëns et al. [19] propose a technique to dynamically compose business processes based on business rules.

III. BUSINESS RULE

According to the Business Rules Group [20], "A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. The business rules that concern the project is atomic - that is, they cannot be broken down further."

In this paper, we use RuleML [16] to represent rules. It's very convenient for service consumers since they don't need to know the detailed XML format of the rules. What they need to do only is to edit the rules in a WSQC (Web Service QoS Constraints) Editor according to their preference or requirements. Then, the editor will automatically transfer the input parameters to a RuleML-based XML file which can be executed by the selection engine. Figure 1 and 2 shows a simple rule with "response time should less than 3000 ms".

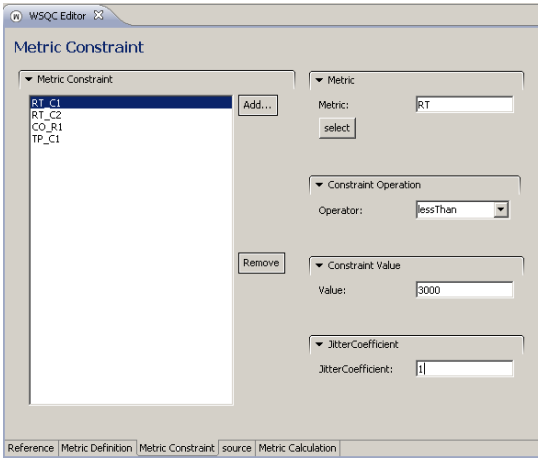

Figure 1. Edit user-defined business rule

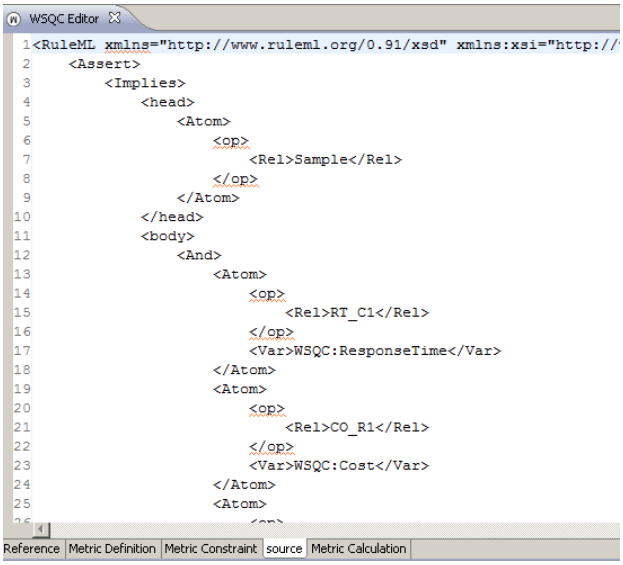

Figure 2. Transfer the user-defined Business Rule into XML Format

IV. FRAMEWORK

Figure 3 depicts a rule-based service composition framework. It composes Web services automatically according to user's functional requirements and predefined rules. The framework is made up of 9 components, which are UDDI business registries, Web Service Crawl Engine, Web Service Repository, Composition Engine, Selection Engine, Rule Engine, Rule Repository, Formula Repository, and the User. The components are described individually as follows.

The UDDI Business Registries (UBRs) are platform-independent, Extensible Markup Language (XML)-based registries for businesses worldwide. They list all kinds of Web services on the Internet. They give businesses a uniform way to describe their services, discover other companies' services, and understand the methods necessary to conduct e-business with a particular company.

The Web Service Crawl Engine (WSCE) [21] was proposed by Eyhab Al-Masri and Qusay H. Mahmoud. It is capable of crawling multiple UDDI Business Registries, and enables for the establishment of a centralized service repository that can be used for discovering Web services much more efficiently. It collects Web services first, and then analyzes Web service information located within the UDDI Business Registry, tModels, and any associated WSDL information. Finally, WSCE stores this information in the WSR.

The Web Service Repository (WSR) is a place to store the information of Web services, such as response time, availability, throughput, successability, reliability, compliance, best practices, latency, documentation, relevancy function, service classification, service name, and WSDL address. The QoSs of services can help SE to best select a Web service to suit user's requirements.

The Composition Engine (CE) is responsible for selecting and filtering resources, and then organize and

assemble them. CE decomposes user requirements, generates abstract process model automatically, and then uses an internal precise search engine to send a SOAP request to the WSR to find the proper services.

The Selection Engine (SE) receives abstract process model and services per node from the CE. Then it interacts with RE to validate user-defined rules and uses NSGA-II to compute the selection. Finally, a detailed description of the composite service is automatically generated and presented to the service requester.

The Rule Repository (RE) stores rules and their properties, including name, creation date, last modification date and associated documentation. It is intended to help achieving rule management.

The Formula Repository (FR) stores the formula for calculating QoSs.

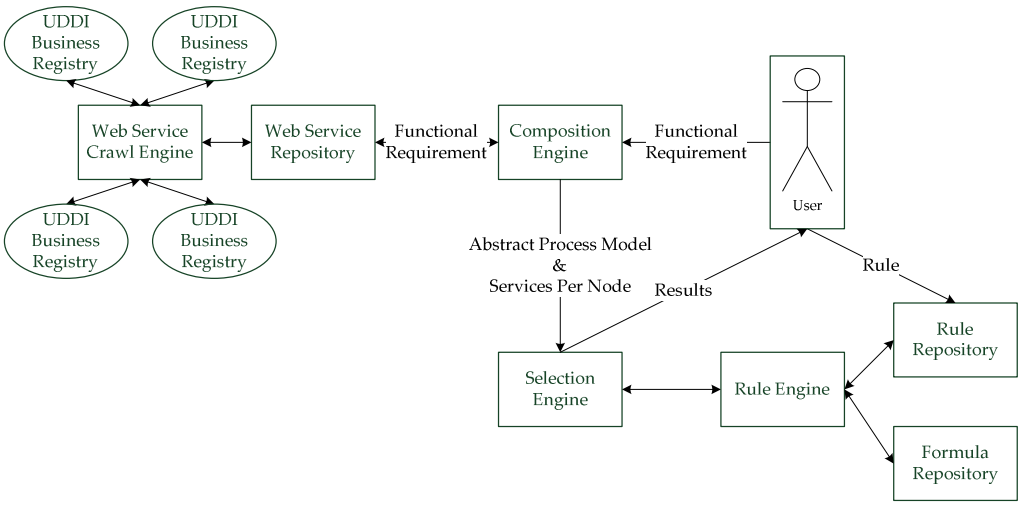

Figure 3. Rule-based Service Composition Framework

V. RULE-BASED SELECTION ALGORITHM

In this section, we elaborate the detailed service composition algorithm's implementation. It combines our previous study and the rule support mechanism.

Figure 4 shows the basic cycle of the rule-based service selection algorithm. The selection algorithm is based on genetic algorithms and integrates non-dominated sorting concept.

A. Definition of Chromosome

In genetic algorithms, a chromosome is a set of parameters which represents a possible solution to the problem. The chromosome is often represented as a string, although a wide variety of other data structures which depend on the application such as integers or real numbers are also used.

In this paper, the chromosome is encoded by an integer array with a number of items such as {1,3,0,2,1,4} as Figure 5 shows. Every gene of the sample chromosome means a choice of Web services, and the values change along with the generations. Genes range from 1 to m_i, where i is the number of abstract service node. From the example, the sample chromosome is equal to the selection "$S_{0,1}$, $S_{1,3}$, $S_{2,0}$, $S_{3,2}$, $S_{4,1}$, $S_{5,4}$".

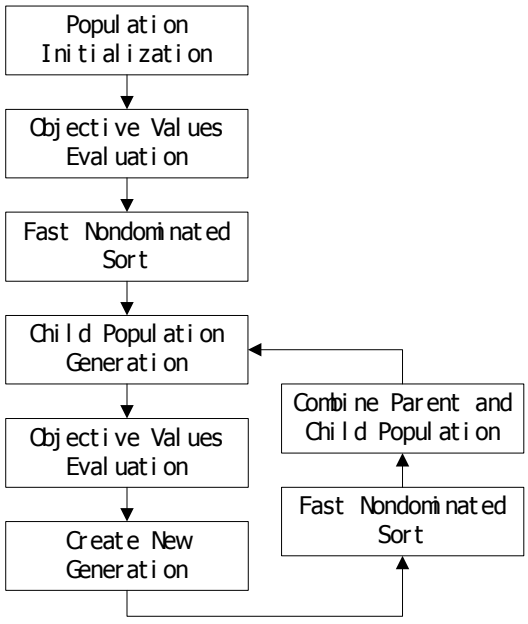

Figure 4. Basic Cycle of Rule-based Service Selection Algorithm

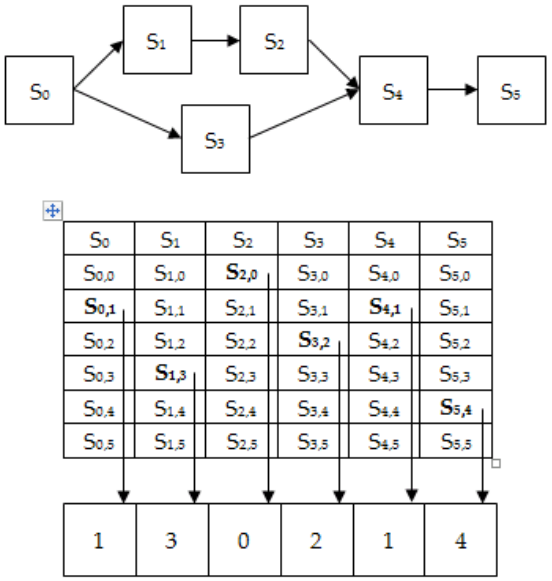

Figure 5. A Sample Chromosome

B. *Produce the Initial Population*

An initial population means the starting point for the genetic algorithm, thus it is usually created randomly.

The size of population indicates the search ability of the algorithm. Small size of the population may cause the quick convergence at early runs and hence limit the search capability. Every coin has two sides, big population size will also lead to the rapid descend in composition speed. From empirical studies, over a wide range of function optimization problems, a population size of between 50 and 150 is recommended. In this paper, we define the population size 120 since the tournament selection operator requires the size be divisible by 4.

C. *Genetic Operators*

Genetic operator consists of three basic operators, i.e., selection, crossover and mutation.

(1) Selection Operator

We use the binary tournament selection as the selection operator because it can obtain better result than the methods of proportional and genitor selection. The binary tournament selection runs a tournament between two individuals and selects the winner.

The domination rule is defined as follows.

An individual A wins the other individual B only in two circumstances. One is that A's rank is higher than B. The other is A's rank is equal to B and A's crowding distance is higher than B.

Here, we add the rule support mechanism. For a certain individual, it will meet some rules and violate others. So the possible situation will be:

1) both A and B meet all rules;

2) A meets all rules and B doesn't or B meets all rules and A doesn't;

3) both A and B doesn't meet all rules.

Situation 1 and 3 are as the original. What affects only is the rank and crowding distance.

In situation 2, rules' satisfaction should be considered. We define the winning rule as: "The one meets all rules will always wins the other even if its rank is much bigger".

(2) Crossover Operator

We use the single-point crossover operator in our experiment. The crossover point's position is based on a random value from 1 to N_t (the number of genes in one chromosome).

We can see the sample crossover process from Figure 6.

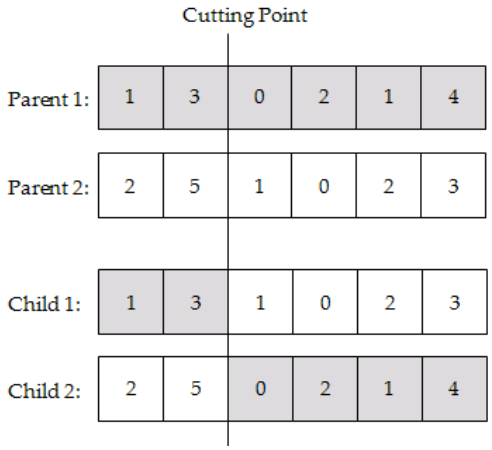

Figure 6. Crossover Operator

(3) Mutation Operator

Mutation operator changes the information contained in the genome according to a given probability distribution. Because a large mutation rate may lead to loss of good solutions and damage the Pareto optimal solution, the mutation rate should be diminished according to the generation count. In this paper, we define the mutation rate 1%.

We can see the sample mutation process from Figure 7.

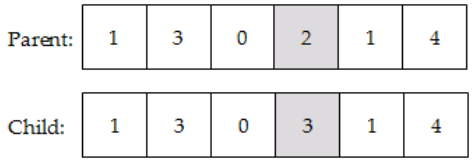

Figure 7. Mutation Operator

(4) Pseudocode

The pseudocode for generate new population is shown in Figure 8.

In step 1, we first create two chromosome arrays a_1 and a_2 which store the generated selection data. In step 2, we shuffle the orders of the arrays a_1, a_2. Then the iteration begins and will last *PopulationSize* times. During the iteration, we use the binary tournament selection operator,

crossover operator, and mutation operator to generate new generation.

```
1. create two chromosome arrays a1, a2
of size PopulationSize with the data
of the last generated chromosomes
2. shuffle orders of a1, a2
3. for i←1 to PopulationSize (i
increase by 4 during each iteration)
  4. do choose parent1 from binary
  tournament of a1[i] and a1[i+1]
  5. choose parent2 from binary
  tournament of a1[i+2] and a1[i+3]
  6. do one point crossover between
  parent1 and parent2 and 2 new
  chromosome are generated
  7. choose parent3 from binary
  tournament of a2[i] and a2[i+1]
  8. choose parent4 from binary
  tournament of a2[i+2] and a2[i+3]
  9. one point crossover between
  parent3 and parent4 and 2 new
  chromosome are generated
  10. mutation
```

Figure 8. Pseudocode of Generate New Population

(5) A Modification

As the entire solution space is limited on the basis of the number of tasks and the max value of the number of services, the algorithm will convergent at early generations if a service is doing crossover with itself. To solve the problem, we add a filter to check and remove duplication chromosomes at the end of each generation.

D. *Evolvement*

The pseudocode of the whole evolvement is shown in Figure 9. It will iterate *GenerationSize* times until it finds the best results.

```
1. for iGen←0 to GenerationSize
  2. do aggregate QoS values
  3. fast nondominated sort
  4. calculate crowding distance
  5. calculate fitness value
  6. generate new population
  7. choose elites to next generation
```

Figure 9. Pseudocode of Evolvement

VI. EXPERIMENTAL RESULTS

In this section, the performance and effectiveness of the proposed algorithm is tested by the experiment. The related parameters are listed as follows.

Experimental related parameters:

Number of tasks: 10;
Number of services for each task: 10;
Number of QoSs for each service: 3;
Size of population: 120;
Size of generation: 100;
Crossover probability: 0.95;
Mutation probability: 0.01.

System environment related parameters:

CPU: Intel Core2 6300 1.86GHz×2
RAM: 2GB
OS: Microsoft Windows XP
JDK: Java SDK 1.6

We first design a pair of experiments to see the effectiveness of the rule-based mechanism. Two selection algorithms are NSGA-II (old) and NSGA-II with the rule support (new). The pre-defined business rule is "overall response time should be less than 200ms". Due to the mechanism for selecting the superior and eliminating, our new algorithm can constantly optimize the population under the guidance of pre-defined rules along the generations. We monitor and record the top ten individuals. Results are shown from Table 6-1 that the rule-based mechanism can prevent chromosomes from violating the rule to a large extent. Six results out of ten meet the requirement while none of the individuals of the old algorithm meet the requirement. From Figure 10, we can see the variation curve of overall response time throughout the entire lifecycle clearly. Individuals which break the rule will be punished due to the mechanism. Thus, it forms an asymptote around 200ms.

TABLE I. TOP TEN CHROMOSOMES

	Overall Response Time(ms)				
NSGA-II without Rule-support Mechanism	434	339	334	405	393
	210	400	502	267	398
NSGA-II with Rule-support Mechanism	76	189	116	129	155
	107	263	216	206	204

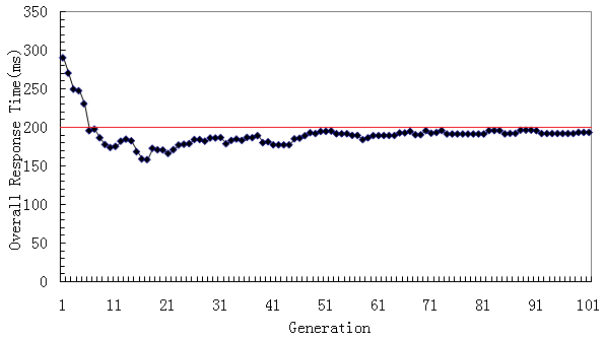

Figure 10. Overall Response Time Tendency of Rule-based WSC (Exp1)

We design a second experiment after rebuilding the services' data. We add another user requirement to see its ability to support multiply rules. The rules come to be "overall response time should be less than 200 ms and overall reliability should be more than 90%". Figure 11 and Figure 12 clearly show the results. We find that results are different from our expectation. Most of the individuals in the last generation don't comply with the first overall response time's rule. We analyze the services' data carefully and find out that the first rule is so strict compared to the services' data that there is no possibility that the composited service can make the overall response time less than 200 ms. However, we see that the rule for availability still works fine and it does return what user wants.

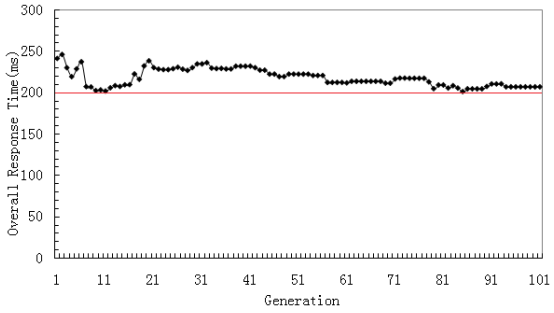

Figure 11. Overall Response Time Tendency of Rule-based WSC (Exp2)

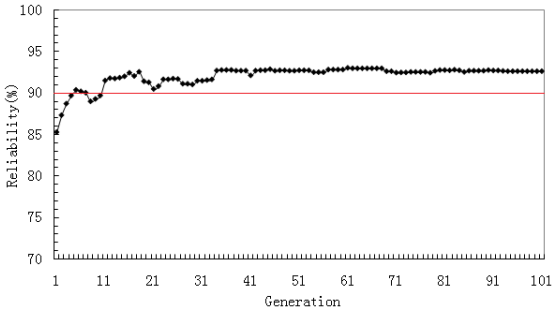

Figure 12. Availability Tendency of Rule-based WSC (Exp2)

VII. CONCLUSION AND FUTURE WORK

This paper presents a novel approach to solve the rule-based Web service composition problem. We propose a composition framework which allows business rules being automatically transferred and executed by the selection engine. An NSGA-II selection algorithm is introduced to get Pereto-optimal solutions. Compared with other selection algorithms such as GA and linear programming algorithms, the NSGA-II enhances the decision support ability and could give service consumers more choices.

Future work will aim to apply the proposed approach to some large-scale service-oriented systems, and to perform a thorough comparison with other non-linear technique. Interactions between the composition engine and the selection engine also need a great deal of research.

ACKNOWLEDGMENT

This paper is supported by the National High-Tech Research Development Program of China (863 program) under Grant No. 2007AA01Z139.

REFERENCES

[1] Web Services Description Language (WSDL) 2.0, http://www.w3.org/TR/wsdl20, 2007

[2] Simple Object Access Protocol (SOAP) 1.2, http://www.w3.org/TR/soap12, 2007

[3] Yujie Yao, Haopeng Chen, "QoS-aware Service Composition Using NSGA-II", The 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, pp. 358-363, 2009

[4] S. R. Ponnekanti and A. Fox, "SWORD: A Developer Toolkit for Web Service Composition," in The 11th International World Wide Web Conference (Web Engineering Track). Honolulu, Hawaii, 2002

[5] F. Casati and M.-C. Shan, "Dynamic and adaptive composition of e-services", The 12th international conference on advanced information systems engineering (CaiSE 00) 2001

[6] H. Sun, X. Wang, B. Zhou, and P. Zou, "Research and Implementation of Dynamic Web Services Composition", Advanced Parallel Processing Technologies (APPT): Springer Berlin / Heidelberg, 2003, pp. 457-466

[7] OASIS Web Services Business Process Execution Language (WSBPEL) Version 2.0, http://docs.oasisopen.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.pdf, 2007

[8] Martin D., Burstein M., Denker G., OWL-S 1.2 Release, http://www.ai.sri.com/daml/services/owl-s/1.2, 2008

[9] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An Approach for QoS-aware Service Composition Based on Genetic Algorithms. In Proc. of Genetic and Computation Conf., June 2005

[10] D. Roman, U. Keller, H. Lausen, R. L. J. de Bruijn, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. Applied Ontology, 1(1):77–106, 2005

[11] Dieter Fensel and Christoph Bussler. The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications, 1(2), 2002

[12] Java Community Process. JSR 94 - Java Rule Engine API. http://jcp.org/en/jsr/detail?id=94, August 2004

[13] Drools 5, http://www.jboss.org/drools

[14] WebSphere ILOG JRules. http://www.ilog.com/products/jrules

[15] JESS, the Rule Engine for the JavaTM Platform, http://herzberg.ca.sandia.gov/jess

[16] RuleML Initiative. http://www.ruleml.org

[17] A. Charfi and M.Mezini. "Hybrid Web Service Composition: Business Processes Meet Business Rules". In Proceedings of the 2nd International Conference on Service Oriented Computing, November 2004

[18] J. Yang and M. P. Papazoglou, "Service Components for Managing Service Composition Lifecycle", Information Sys-tems, vol. 29, pp. 97-125, 2004

[19] B. Orri¨ens, J. Yang, and M. P. Papazoglou. "A Framework for Business Rule Driven Service Composition". In Proceedings of the Fourth International Workshop on Conceptual Modeling Approaches for e-Business Dealing with Business Volatility, 2003

[20] The Business Rules Group, "Defining Business Rules, What are they really?" www.businessrulesgroup.org, July 2000

[21] E. Al-Masri, and Q. H. Mahmoud, "WSCE: A crawler engine for large-scale discovery of web services", ICWS pp.1104-1111, 2007