Personalized Reliable Web service Compositions

Daniela Barreiro Claro[1], Oriana Licchelli[2], Patrick Albers[2], and Raimundo Jose de Araujo Macedo[1]

[1] LASID/DCC/UFBA, Federal University of Bahia,
Av. Adhemar de Barros, s/n, Salvador, Bahia, Brazil
{dclaro, macedo}@ufba.br
[2] ESEO, 4 rue Merlet de la Boulaye
BP30926 49009 - Angers, France
oriana.licchelli@essca.fr, patrick.albers@eseo.fr

Abstract. Nowadays Internet is anywhere and users can find all possible information. In this situation, the new challenge is to provide the right information at the right time. As Web services are becoming a de facto way to integrate heterogeneous applications, companies are publishing some of their functionalities using a Web service format. Consequently, users can search for Web services so as to resolve complex tasks. These tasks, on the other hand, are not customized for a specific user. This can leverage an amount of information unnecessary to a user profile. Thus, we propose an implemented framework (SAREK) to automatically compose semantic Web services by means of personal profiles. SAREK interacts with an ontology in order to discover semantic Web services and uses a planning algorithm to determine all tasks that belong to the composition. Moreover, this implemented framework ensures reliable composition executions respecting such user profiles.

Keywords: Semantic Web service, Web service composition, user profile, reliable compositions.

1 Introduction

The number of services published on the Internet as a Web service format has increased tremendously last years. In some situations, a single Web service cannot fulfill a user request, thus, it is necessary to compose them; this process is called *Web service composition*. As Web service is an autonomous and self-contained application, it is hard to manage automatically its composition. Thus, many researchers have proposed a framework in order to manage the life-cycle of a Web service composition.

In this work, we propose to enhance the SPOC[1] framework with user profiles and reliable compositions. SPOC is an implemented framework to automatically compose Web services with a minimum human intervention. Users just give a request and the framework tries to find compositions that reaches this request; then these compositions are presented to the user.

The enhanced SPOC, now called SAREK[3], tries to overcome some detected drawbacks: the planning phase that used a JESS algorithm and did not re-plan; the set of tradeoff compositions that gave to the user a list of possible compositions with no order and no preference; and the Web service execution that did not exist in the previous version.

Thus, the main advantages of SAREK are:

- a simple planning algorithm based on preconditions and effects namely GPA (Goal-oriented Planning Algorithm) that allows re-planning,
- a new user profile-based composition choice procedure that allows customizing compositions and,
- a system re-engineering to achieve better reliability execution.

In order to validate its implementation, SAREK framework was applied to a bidding process via Internet, a kind of e-Government application.

The main objective of this paper is to present SAREK framework, highlighting the Goal-oriented Planning Algorithm (GPA) and its interaction with an ontology and the user profile mechanism (a recommendation system). The remaining of this paper is organized as follows. The next section overviews the framework SAREK. The third section details the Planner module of SAREK and explains the GPA algorithm. The fourth section focus on personalized compositions. The fifth section presents how to obtain reliable composition executions issued from user profiles. The sixth section describes the case study (e-government application). Before concluding, the last section presents related work and points to some future work.

2 Web Service Composition Framework

Web service composition (WSC) does not only mean putting Web services together but actually taking into account the relations (based on preconditions and effects) among these services and coordinate them to fulfill a given user goal. There are two main concepts in SAREK : *activity* and *Web service*. An activity[4] is a kind of abstract Web service that describes different functionalities. Web services in SAREK are considered as an implementation of a given activity, i.e. a Web service performs an activity. Consequently, we can associate an activity to a set of candidate Web services that have semantic similar functionalities. In SAREK, all Web services must be an instance of an activity.

SAREK is divided into two major modules: the *Planner Module* and the *Executor Module*. First of all, the user defines the goal and different inputs. The Planner Module aims firstly to determine which activity should be incorporated into the composition in order to fulfill the user request, and secondly find a set of candidate Web services for each activity. The Executor Module aims to execute

[3] In the fictional Star Trek universe, Sarek is a Vulcan ambassador, and father of Spock.

[4] In this paper the word *activity* and *task* are used interchangeable.

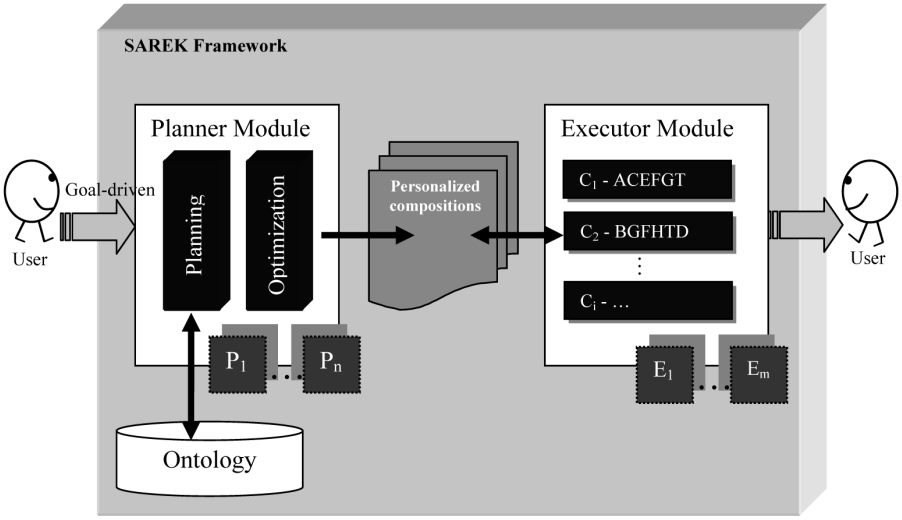

Fig. 1. SAREK Framework

the selected composition. This composition can be selected directly by the user or automatically by an algorithm based on user profiles. The figure 1 depicts SAREK general framework.

3 Planned Compositions

The Planner Module determines the set of activities which should be included in the plan so as to fulfill the given user goal. SAREK starts by searching for activities inside an ontology repository that treats semantically similar concepts. This repository describes the application domain (e-government) using an OWL format. Each individual in this ontology denotes a task and each task can be performed by a set of published Web services (candidate Web services). Thus, SAREK can find semantic similar Web services to perform a task. All Web services in SAREK are described using an OWL-S [2]. OWL-S (S stands for services) describes a Web service in unambiguous manner and also includes its preconditions and effects. In SAREK, this OWL repository is called OPS (Ontology to Publish Services). The notion of task/activity (work) and service (company) has already been explored by others authors [3–6].

In order to determine which activity belongs to a composition, we developed a planning algorithm called GPA (Goal-oriented Planning Algorithm). This is a simple algorithm based on preconditions and effects [7, 8]. Composition problems are composed of three components: the user request, the Web services and the initial user parameters. These three components can be represented in planning domain respectively as the goal, the actions and the initial states.

GPA Algorithm:

1: receive g given by the user
2: match e_i with g
3: find p_i of e_i in action a_i
4: **for** each p_i **do**
5: match effect e_i with p_i
6: *actionList* = a_i
7: **end for**
8: **if** p_i **is null**
9: show plan of actions *actionList*

Fig. 2. GPA Algorithm

The GPA algorithm starts with the goal g given by the user. Therefore, it searches all actions that match its effects e_i with the given user goal g. When action a_i is found, GPA retrieves its preconditions p_i. For each precondition p_i, GPA matches the effects and increases the *actionList* with a_i. GPA ends when no more preconditions are found. Afterwards, the *actionList* is presented. In order to search for new actions, GPA algorithm interacts with the OPS (ontology to Publish Services) to retrieve new tasks.

Once the *actionList* has all actions, the next phase in the Planner Module is *Optimization*. SAREK optimizes the compositions based on a multiobjective algorithm. As a result of this phase, a set of semantic similar compositions are given to the user. In multiobjective problems, objectives can be conflicting, for example, minimize cost and maximize reputation. In such situations, the notion of optimal solution is generalized to the notion of Pareto optimal solutions. The results of the *Optimization* phase is a set of Pareto solutions, that in SAREK it represents a set of tradeoff compositions [9].

SAREK uses a multiobjective evolutionary algorithm (MOEA)[9], which is among the most powerful methods for multiobjective optimization[10]. Such an algorithm is able to produce a set of (nondominated) solutions for each run of the algorithm based on its populations. More specifically, SAREK uses a genetic algorithm called NSGA-II [11]. Once a set of semantic similar compositions is obtained, the user might choose one solution to be executed. Instead of choosing randomly a composition, we propose to classify the set of compositions based on user profiles.

4 Personalized Compositions

As stated previously, the result of optimization phase is a set of compositions. In each composition, the services are selected according to estimated values and thus only define a general and impersonal evaluation. It would be possible to take user preferences into account to obtain a personalized presentation of the candidate services compositions. It is the reason why the list of results of the

optimization phase is given to the personalization phase (a recommendation system).

4.1 Recommender Systems

Recommender systems can be defined as systems that produce individualized recommendations as outputs or have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible solutions. Recommender systems learn about a person's needs and interests and then identify and recommend information that meets those needs. Recommender Systems consist of three major components [12]:

1. *Background data*, the information that the system has before starting the recommendation
2. *Input data*, information the user must supply to the system in order for a recommendation to be made
3. *Algorithm*, which combines the background and input data to arrive at a suggestion

Although various types of recommender systems have been proposed, their filtering techniques fall mainly into two categories: Content-based filtering and Collaborative filtering. Our recommendation system is based on user's features and rates to order, in base to composition rate, the result list of optimization phase of SAREK system and presenting it to the user. Then the user can select a Web service composition from this list. The recommendation system use a Gaussian Bayes Classifier that is based on Bayes rule to compute the probabilities, which are used in order to make predictions. This system permits to predict the rate r that a user u will give for a target service s using the traditional Bayes classifier [13] with the Gauss distribution. The predicted rate and the features of web service will be used for writing a Horn clause (rule) in the profile of the user u.

4.2 How to use user profiles in SAREK

In SAREK system, each candidate Web service has the following set of estimated values:

- *Cost* (the budget to perform the service task)
- *Turnover* (the ratio of annual sales of the service provider)
- *Execution time* (the time spent by the service provider to perform a task)
- *Reputation* (measure of service trustworthiness for performing a given task)

The user profile is stored in a RDF file. It contains user's data, different used services with its user rate real or predicted with Gaussian Bayes classifier, and a set of Gaussian Bayes rules. The profile data are represented as follow:

Profile =

- *personal data (name, login, ...)*
- *activities data:*
  - *rules for Gaussian Bayes classifier*
  - *rates (real or predict) for each service*

The user employs our recommender system to create his/her first profile with personal data and a set of basic rules for Gaussian Bayes classifier. Then, in every time, he/she can rate the web services with a rate among 1 to 5 where 1 is very bad and 5 is very good. This rate is stored in his/her profile as a Horn clause that contains the quality parameters and the rate of the evaluated service. For instance, there is the *Joao supply cement* service with the following quality parameters:

- reputation is 6.12
- cost is 197.82
- time is 0.3
- turnover is 3.456E9

If the user James rates this service with the value 4 then the system stores this information in his profile like the following rule:

$((0 < reputation <= 6.12) and (0 < cost <= 197.82) and (0 < time <= 0.3) and (0 < turnover <= 3.456E9)) ==> 4$

specifying the type of the service (supply cement service) but without specifying the service name (*Joao supply concrete* service).

5 Reliable Compositions

As personalized compositions are ordered and presented in the Planner Module, the Executor Module executes the first composition following the best user profile ranking. The Executor module executes the composition defined by the user profile activating alternative execution paths when necessary (due to failure of composite services). Reliable compositions in SAREK are achieved in two steps: a passive replication technique for each module (Planner and Executor) and a semantic replication scheme for each composition.

Both modules are replicated using a passive replication technique. If the primary module fails, a backup is voted and takes over the execution. There are $n, n \geq 2$ instances of the Planner Module, where a $P_i, i \in [1..n]$ is voted the primary Planner Module and the others will act as backups. The backup Planner Module (or the set of backup Planner Modules) starts at the same time as the primary one. The actual execution of all backup Planners monitors P_i's execution. If P_i, for any reason, fails a new P_i is chosen and then it takes over the execution. This process ensures that even if the Planner module crashes, SAREK is still able to fulfill the user request. It is important to observe that

if the Planner does not accomplish its goal, the Executor module cannot even start. Executor Module follows the same replication technique.

The semantic replication scheme takes over on each composition. During execution of a composition, faulty Web services can be replaced by semantically similar services. We argue that this is a kind of spatial redundancy because there is a set of compositions that achieve the same goal ordered by user profiles. This scheme uses a prefix approach so as to increase performance when re-executing a partially failed composition. The prefix algorithm works as follows. Assume the running composition is defined as $< s_1, s_2, s_3, ..., s_n >$ and that this composition fails because of the failure of s_3, but services s_1 and s_2 were run correctly. Then, the prefix algorithm searches another composition that starts with the prefix $< s_1, s_2 >$ in order to save recovery time. A preliminary evaluation of the SAREK's dependability mechanisms has been presented elsewhere[14].

Thus, these reliable mechanisms and techniques of user profiles guarantee the execution of customized reliable compositions in SAREK framework.

6 Case Study - Bidding Process for Public Buildings

Our case study focuses on bidding process for public buildings via Internet. Old public buildings always need to be repaired.

The bidding process for repairing public buildings starts with a request for restoration. As regards this request, an architect with a functionary determines all general work that have to be carried out in the public building. These works are grouped into categories based on activity. The bidding process is organized by category such as Electricity, Masonry, etc.

Afterwards, the architect defines in which order the works have to be performed. Once the work plan is determined, companies can send their propositions with estimated costs.

Once propositions have been received, the functionary analyzes them, one by one, based on their costs, duration of work, company's turnover and reputation so as to find a good combination between company and work.

In order to make a good and interesting combination between company and work, the functionary should simultaneously take into consideration several criteria such as cost, duration, turnover and reputation. Such a multicriteria analysis leads the functionary to decide which company performs which work and the bidding process terminates.

Applying SAREK to this case study, work is an activity and company is a Web service.

A Scenario with user profiles and reliable executions

The user John Smith, a functionary, is the supervisor for building a concrete staircase in a public building. He starts a bidding process to find a set of companies to carry out this work. John has to compare each company manually or can use a SAREK-based application. Using SAREK-based application, John should define his goal ("build a concrete staircase") and the application

finds all works to fulfill John's request: "supply concrete" and "build a concrete staircase". Moreover, SAREK also indicates the order that these works have to be carried out. This work order indicates that supply concrete is a precondition of building a staircase. At this moment, the functionary can choose whether to obtain a general or a customized result.

If John chooses to have a general result (see Figure 3), he can select a solution scrolling down the given list of company compositions. Indeed, if the proposed list is long, the task of selecting a good result can consume time and effort.

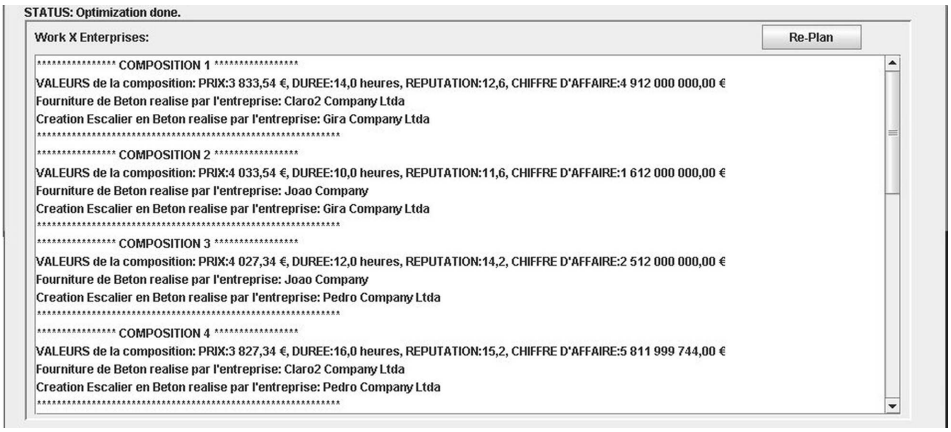

Fig. 3. SAREK system: result optimization phase

However, if John chooses a personalized result, he has to create his profile in the recommender system. To create his profile, John has to use the interface of our recommender system. The system generates a profile with personal data and a set of basic rules for the Gaussian Bayes classifier. Therefore, John can insert his rate for each Web service (company) in OPS (" e-government ontology"). For instance, John rates 2 for the Web service ("supply concrete') provided by the company "Joao Company" (see figure 5). Afterwards, the application starts the optimization procedure to find all distinct solutions and then exploits John's profile to rank the results (compositions).

Figure 4 depicts a new rank for compositions. For instance, the third composition in Figure 4 was actually the first composition in general optimization phase (without user profiles) in Figure 3. Of course, John can disagree with the set of solutions proposed by SAREK (with user profile). Thus he can choose another solution that is not at the first position. In this situation, the functionary may use again the interface of the recommender system to modify the rate of services in his profile (as described in the section 4.2) for improving future uses of SAREK.

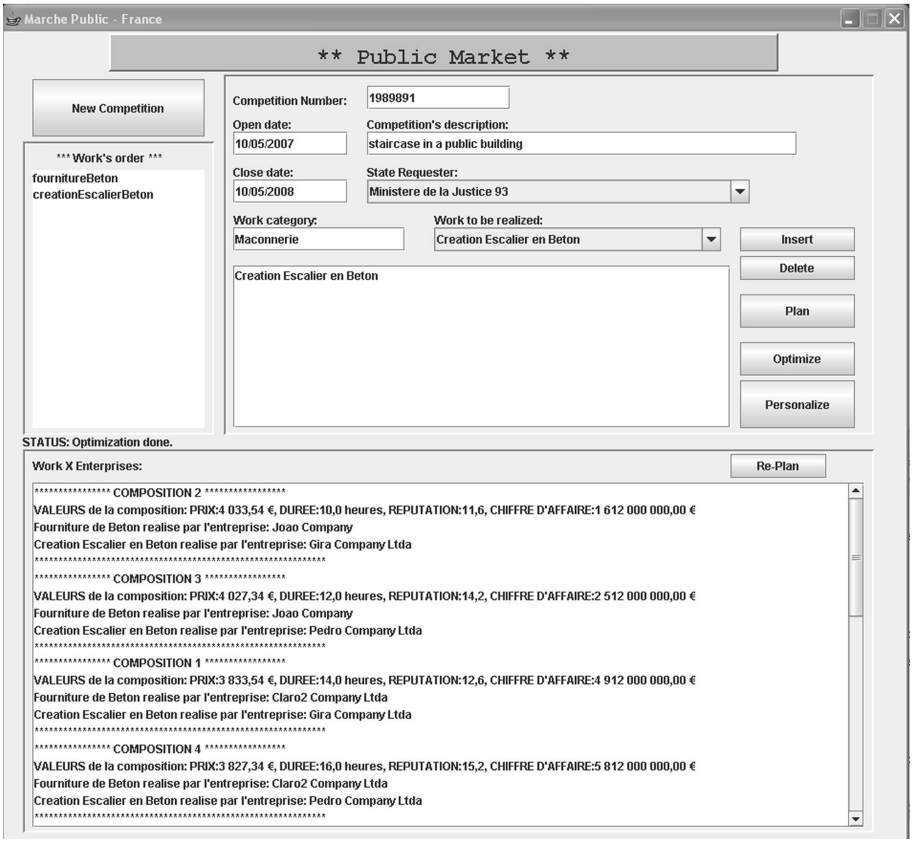

Fig. 4. Personal result of the SAREK system for the user John

Once John has chosen his composition, the Executor Module starts trying to perform the composite companies (Web services). As regards the above example: "supply concrete" and "build a concrete staircase", suppose that the enterprise "BnB Company" performs the "supply concrete" activity but its Web service has failed by crash. By using our prefix technique, SAREK tries to execute another semantic similar composition, which starts with the same prefix, i.e. "supply concrete". Respecting the user profile ranked compositions, SAREK searches the next following composition in the user profile list. If all these fault tolerant mechanisms of SAREK fail in executing a composition, an error is shown to the user informing that the execution of the Web service composition was not possible.

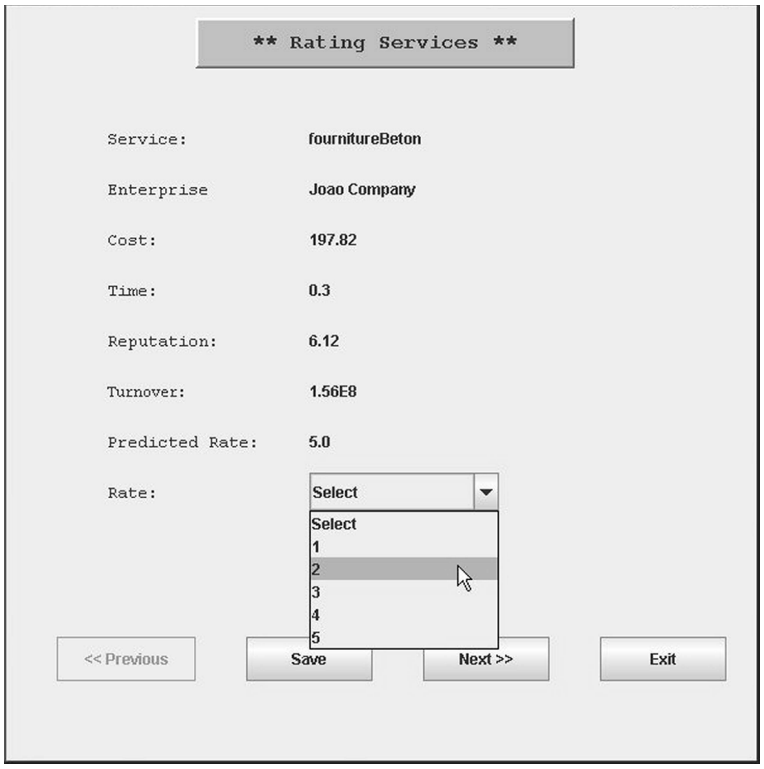

Fig. 5. SAREK system: service rate

7 Related Work

Several authors have studied the possibility to provide personal Web service compositions. In particular, [15] have developed a prototype system to provide user-driven Web service compositions. At each phase of the composition the system asks the user what Web services wanted. [16] have developed a system which chooses Web services automatically using the Item based collaborative filtering approach.

On the other hand, researchers have also been studying solutions for reliable compositions. In particular, authors in [17, 18] propose transactional mechanisms in compositions. The authors in [19] propose a replication mechanism based on group communication. However, none of them deals with reliable Web service compositions. Our proposition is to use SAREK to automatically compose Web services and to exploit the user profiles in order to present the most suitable solutions ensuring a reliable execution of the selected composition.

8 Conclusion

The incorporation of planning algorithms as well as Semantic Web, in particular ontologies, have leveraged new research challenges in Web service compositions. SAREK incorporates some of these challenges in both modules: using the GPA algorithm interacting with an ontology to determine the set of activities (Planner Module) and proposing a replication scheme based on semantic similar compositions in the presence of faulty services (Executor Module). As far as information published on the Internet increases, a customized solution based on user profile should be incorporated into applications. SAREK uses a recommendation system (Gaussian Bayes classifier) in order to propose the most suitable solutions for the user. Thus, SAREK ensures reliable executions of Web service compositions taking into account user profiles to propose a set of customized compositions.

This approach has been validated using a French e-government ontology to restore public buildings. As future work, we will carry out an experiment in a real Web context with a set of real users. The goal of this experiment will be to compare SAREK results (a list of Web service compositions) with and without user profile in order to evaluate the user satisfaction.

9 Acknowledgments

Dr. Daniela Barreiro Claro is supported by FAPESB(BOL2071/2006) and Prof. Raimundo Jose de Araujo Macedo is supported by FAPESB and CNPQ (Edital Universal).

References

1. Claro, D.B., Albers, P., Hao, J.K.: A framework for automatic composition of rfq web services. In: IEEE Proceedings of the First Workshop on Web Service Composition and Adaptation (WSCA) held in conjunction with International Conference of Web Services (ICWS'07), Salt Lake City, Utah, USA (2007) 221–22
2. Coalition, O.: Owl-s: Semantic markup for web services (2006) http://www.daml.org/services/owl-s/1.1/.
3. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. on Software Engineering (2004)
4. Motta, E., Domingue, J., Cabral, L., Gaspari, M.: Irs-ii:a framework and infrastructure for semantic web services. In: Proceedings of International Semantic Web Conference (ISWC 2003), Florida, USA (2003)
5. Maamar, Z., Sheng, Q.Z., Benatallah, B.: On composite web services provisioning in an environment of fixed mobile computing resources. Information Technology and Management **5** (2004) 251–270
6. Aggarwal, R., Verma, K., Miller, J., Milnor, W.: Constraint driven web service composition in meteor-s. In: Proceedings of IEEE SCC 2004. (2004) 23–30
7. McIlraith, S., Son, T.C.: Adapting golog for composition of semantic web services. In: Proceedings of the Conference of Knowledge Representation and Reasonning. (2002)

8. Pistori, M., Traverso, P., Bertoli, P.: Automated composition of web services by planning in asynchronous domains. In: Proceedings of 15^{th} International Conference on Automated Planning and Scheduling (ICAPS 2005), Monterey (2005) 2–11
9. Coello, C.C.A., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic, New York (2002)
10. Claro, D.B., Albers, P., Hao, J.K.: Chapter 8: Selecting Web Services for Optimal Compositions. In: Semantic Web Services, Processes and Applications. Springer (2006) 203–233
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: Nsga-ii. IEEE Trans Evol Computat **6** (2002) 182–197
12. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modelling and User-Adapted Interaction **12** (2002) 331–370
13. Langley, P., Iba, W., Thompson, K.: An analysis of bayesian classifiers. In: Proceedings of the 10^{th} National Conference on Artificial Intelligence, San Jose, CA, AAAI Press (1992) 223–228
14. Claro, D.B., Macedo, R.J.A.: Dependable web service compositions using a semantic replication scheme. In: Proceedings of XXVI Brazilian Symposium of Networks and Distributed Systems(SBRC2008), Rio de Janeiro,RJ, Brazil (2008) 441–454
15. Sirin, E., Parsia, B., Hendler, J.: Filtering and selecting semantic web services with interactive composition techniques. IEEE Intelligent Systems **19** (2004) 42–49
16. Manikrao, U.S., V.Prabhakar, T.: Dynamic selection of web services with recommendation system. In: Proceedings of International Conference on Next Generation Web Services Practices (NWeSP 2005), Seoul, Korea (2005)
17. Gorbenko, A., Kharchenko, V., Romanovsky, A.: On composing dependable web services using undependable web components. Int. J. Simulation and Process Modelling **3** (2007) 45–54
18. Pires, P.F., Benevides, M.R.F., Mattoso, M.: Building reliable web service composition. In: NODE'2002 Web and Database - related Workshops on Web, Web-Services and Databases Systems. Volume 2593., London,UK, LNCS (2002) 59–72
19. Salas, J., Prez-Sorrosal, F., Patino-Martinez, M., Jimenez-Peris, R.: Ws-replication: A framework for highly available web services. In: WWW'2006 - International World Wide Web Conference, Edinburgh, Scotland, ACM (2006) 357–366