Advance BPEL execution adaptation using QoS parameters and collaborative filtering techniques

Dionisios D. Margaris*

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
margaris@di.uoa.gr

Abstract. In this thesis, frameworks for providing runtime adaptation for BPEL scenarios are proposed. The adaptation is based on (a) quality of service parameters of available web services (b) quality of service policies specified by users and (c) collaborative filtering techniques, allowing clients to further refine the adaptation process by considering service selections made by other clients.

1 Introduction

Web Services are considered a dominant standard for distributed application communication over the internet. Consumer applications can locate and invoke complex functionality, through widespread XML-based protocols, without any concern about technological decisions or implementation details on the side of the service provider. Web Services Business Process Execution Language (WS-BPEL) [1] allows designers to orchestrate individual services so as to construct higher level business processes; the orchestration specification is expressed in an XML-based language, and it is deployed in a BPEL execution engine, made thus available for invocation by consumers. -BPEL has been designed to model business processes that are fairly stable, and thus involve the invocation of web services that are known beforehand.

In this dissertation an adaptation algorithm uses both QoS specifications and semantic-based collaborative filtering personalization techniques to decide which offered services best fit the client's profile is presented. To achieve this goal, the metasearch algorithm paradigm [4] is followed, using two different candidate adaptation ranking algorithms, the first examining the QoS aspects only and the second being based on collaborative filtering techniques. The adaptation rankings produced by these two algorithms are combined to generate the overall ranking, which then drives the adaptation. The combination of the results is performed using a weighted metasearch score combination algorithm ([4][5]), however varying weights are used to address issues associated with collaborative filtering, such as cold start (i.e. few entries recorded in the rating database, thus no good matches can be obtained) and

*Dissertation Advisor: Panayiotis Georgiadis, Emeritus Professor gray sheep (i.e. unusual users, which cannot be matched with other users even after the database has been adequately populated).

The adaptation is based on (a) quality of service parameters of available web services (b) quality of service policies specified by users and (c) collaborative filtering techniques, allowing clients to further refine the adaptation process by considering service selections made by other clients.

The combined proposed BPEL execution framework includes provisions for

(a) specifying QoS requirements for invocations of web services within a WS-BPEL scenario
(b) specifying specific bindings for selecting services and designating which services are subject to adaptation,
(c) adapting the WS-BPEL scenario execution according to the results of the service selection algorithm,
(d) monitoring the behavior of the invoked services regarding their QoS aspects,
(e) collecting user satisfaction feedback about the invoked services and taking these data into account when formulating recommendations and
(f) caters maintaining the transactional semantics that invocations to multiple services offered by the same provider may bear.

This approach follows the horizontal adaptation paradigm since, as noted in [2], horizontal adaptation preserves the execution flow which has been crafted by the designer to reflect particularities of the business process, while it also allows the exploitation of specialized exception handlers.

2 Related Work

As stated above, existing adaptation approaches follow either the horizontal or the vertical adaptation approach. [4] performs horizontal QoS-based adaptation, taking into account the sequential and parallel execution structures within the BPEL scenario. Note that none of above approaches incorporates CF techniques to enhance the quality of the adaptation. [5] presents CF techniques to drive the adaptation, and an associated execution framework, it however uses very limited QoS-based criteria (only a lower and an upper bound for each QoS attribute), hence it runs the risk of formulating solutions whose QoS is much inferior to the optimal composition QoS that can be attained, especially in cases that CF has known issues (e.g. cold start and gray sheep). In order to perform QoS/CF-based adaptation or exception resolution, all approaches employ means to formally specify the services' functionality; techniques involving QoS characteristics need additionally to have access the services' QoS attribute values. In this work, we adopt the subsumption relationship approach [18] due to its expressiveness and flexibility. An important aspect of QoS attributes is that their values may vary, according to server load, network conditions or other relevant factors. To this end, work in [19] is adopted to allow a more accurate estimation of QoS attribute values; this increased accuracy can be used in adaptation systems to improve the quality of the adaptation.

3 QoS AND CF UNDERPINNINGS

In the following subsections we summarize the underpinnings from the areas of QoS and CF, which are used in our work.

3.1 QoS concepts

For conciseness purposes, in this paper we will consider only the attributes responseTime (rt), cost (c) and availability (av), adopting their definitions from [9]. This does not lead to loss of generality, since the algorithms can be straightforwardly extended to accommodate more attributes. The QoS specifications for a service within the BPEL scenario may include an upper bound and a lower bound for each QoS attribute, i.e. for each service sj included in a BPEL scenario, the designer formulates two vectors MINj=(minrt,j, minc,j, minav,j) and MAXj=(maxrt,j, maxc,j, maxav,j). Additionally the designer formulates a weight vector W=(rtw, cw, avw), indicating how important each QoS attribute is considered by the designer in the context of the particular operation invocation. The values of the QoS attributes are assumed to be expressed in a “larger values are better” setup, e.g. a service having cost = 6 means that that it is cheaper than a service having cost = 4. In order to compute the QoS attribute values of a service S composed from constituent services s1, …, sn having QoS attributes equal to (rt1, c1, av1), …, (rtn, cn, avn), respectively, the formulas given in table 1 [10] can be used. Note that these formulas do not take into account the possibility that some service is executed multiple times within a loop or conditionally with a probability of p; these aspects will be considered in our future work.

	QoS attribute		
	responseTime	cost	availability
Sequential composition	$\sum_{i=1}^{n} rt_i$	$\sum_{i=1}^{n} c_i$	$\prod_{i=1}^{n} av_i$
Parallel composition	$\max_i(rt_i)$	$\sum_{i=1}^{n} c_i$	$\prod_{i=1}^{n} av_i$

Table 1. QoS of composite services

Most works dealing with QoS-based BPEL scenario execution adaptation, consider given QoS attribute values for each service, which can be for instance declared by the service provider within an SLA. However, in the real world, QoS metrics such as response time and availability may vary, due to server or network conditions (failures, overloads, bottlenecks etc). To tackle this issue, in this paper, we employ prediction models for QoS attribute values, in order to use in the recommentation process values that are closer to the actual ones, improving thus the accuracy of the adaptation. In particular, we adopt [7] and [8] for predicting the service response time and service availability, respectively. Both these algorithms predict future performance of services by examining past measurements; the platform proposed in this work collects these measurements when invoking services in the context of BPEL scenario executions and makes them available to the modules predicting the future QoS values.

3.2 Subsumption relationship representation

In order to adapt the BPEL scenario execution, the adaptation engine needs to be able to find which services offer the same functionality, and are thus candidate for invocation when this particular functionality is needed. In this work, we represent this information using subsumption relationships [6] which, for any pair of services S1 and S2 defined as follows: (i) S1 exact S2, iff S1 provides the same functionality with S2 (ii) S1 plugin S2, iff S1 provides more specific functionality than S2; in this case S1 could be used whenever the functionality of S2 is needed, since it delivers (a specialization of) the functionality delivered by S2 (iii) S1 subsume S2, iff S2 provides more generic functionality than S2. In this case S1 cannot unconditionally be used whenever the functionality of S2 is needed and (iv) S1 fail S2, in all other cases; in this case, S2 cannot be substituted for S2. Under these definitions, a service A can be unconditionally substituted by a service B if (A exact B or A plugin B); this setup provides more flexibility as compared to strict service equivalence (A exact B) regarding the formulation of the adapted execution plan, and is hence adopted in this paper. Effectively, subsumption relationships organize services in a tree, where generic services are located towards the root and more specific services towards the leaves [6]. Tree nodes, besides service identity, can accommodate QoS values for the services they represent; this information can be stored in repositories such as OPUCE [11]. Fig. 1 shows an excerpt of a subsumption relationships tree.

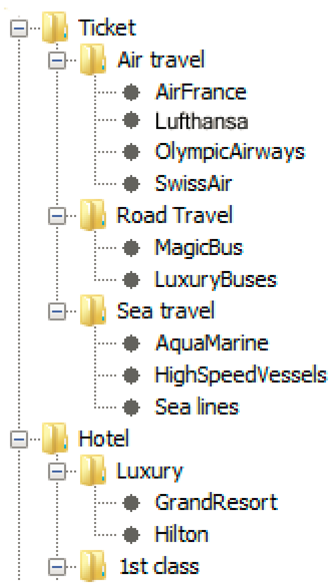

Fig. 1. Example subsumption relationships tree

3.3 Designations on specific service bindings and functionality omissions

As noted above, users may wish to designate exact services to be invoked for realizing specific functionalities, while asking for recommendations on other ones. For instance, in a travel planning scenario the consumer may request that s/he travels by “Sea Lines”. Further, the consumer may also specify that some functionality optionally included in the BPEL scenario is not executed; for example, a tourist may not want to rent a car, while such a provision is present in the scenario. Typically, the BPEL code will examine input parameters and decide using a conditional execution construct (<switch>) whether to invoke the functionality or not. Finally, functionalities that are neither explicitly bound to specific services, nor are designated as “not to be executed” are subject to adaptation. We consider that specific bindings and designations for functionality omissions are explicitly expressed in the request for scenario invocations.

3.4 Usage patterns repository

In order to perform CF-based adaptation, a repository with user ratings for services is required. In this paper, we adopt the representation used in [5], where the ratings repository is modeled as a table having a number of columns equal to the functionalities present in the BPEL scenario, and one row for each BPEL scenario execution. Cell i,j is filled with value S if during the ith execution of the BPEL scenario, service S was used to implement functionality j; cell (i, j) may be also blank, if during the ith execution of the BPEL scenario functionality j was omitted. In order to accommodate user ratings, we extend this repository by adding one column per functionality. This column stores an integer value from the domain [1, 10], corresponding to the rating given by the user that executed the particular scenario instance. For the cases that the user has not provided a rating, a null value is stored and the CF-based algorithm uses a default value, as explained in section 4. The BPEL scenario adaptation module inserts new records to the usage patterns repository (UPR), when the concrete services that will be invoked in the context of a particular BPEL scenario execution are decided, while the user evaluation collection module arranges for storing user rankings in the relevant columns. Table 2 presents an example UPR.

# exec	Travel	Hotel	Event
1	OlympicAirways	YouthHostel	ChampionsLeague
2	SwissAir	Hilton	GrandConcert
3	HighSpeedVessels	YouthHostel	
4	LuxuryBuses		EuroleagueFinals
5	Lufthansa	YouthHostel	GrandConcert
6	AirFrance	Hilton	
7	SwissAir	YouthHostel	ChampionsLeague

Table 2. Example usage patterns repository

4 THE SERVICE RECOMMENDATION ALGORITHM

As stated in section 1, our approach follows the horizontal adaptation paradigm, leaving the composition logic intact and adapting the execution by selecting which concrete service implementation will be used in each specific invocation. To perform this task, the algorithm takes into account the following criteria:

- The consumer’s QoS specifications (bounds and weights).
- Designations on which exact services should be invoked, if such bindings are requested by the consumer (e.g. a user wanting to travel using Air France).
- Designations on which functionalities should not be invoked
(e.g. a user wanting to book a trip without scheduling any event attendance).
- The QoS characteristics of the available service implementations, including monitored values of the QoS attributes of the services.
- The service subsumption relationships.
- The UPR, which includes user ratings.

The approach proposed in this paper incorporates two different candidate service ranking algorithms, the first examining the QoS aspects only ([4]) and the second being based on CF techniques ([5]). The algorithms run in parallel to formulate their suggestions regarding the services that should be used in the adapted execution, and subsequently their suggestions are combined, through a metasearch score combination algorithm with varying weights.

4.1 The QoS-based adaptation algorithm

The QoS-based adaptation algorithm initially identifies the services which are candidate to be used for delivering functionalities in the context of the current BPEL scenario, respecting the QoS-bounds set by the user, and subsequently computes the k-best service assignments to the functionalities requested for the particular scenario execution. In more detail, the algorithm proceeds as follows:

- For each functionality f_i for which adaptation has been requested, the algorithm retrieves from the semantic service repository the concrete services that (a) deliver this functionality and (b) respect the QoS bounds set by the users. These are the candidates for implementing functionality f_i. Formally, this is expressed as

$$CF(i) = \{s \in Repository: (funct_i\ exact\ s\ \vee funct_i\ plugin\ s)\ \wedge\ [(min_{rt,i} \leq rt_s \leq max_{rt,i})\ \wedge\ (min_{c,i} \leq c_s \leq max_{c,i}) \wedge\ (min_{rel,i} \leq rel_s \leq max_{rel,i})]\}$$

Note that in all steps of this algorithm, the QoS values for response time and availability considered for each service are those returned by predictor methods [7] and [8], respectively.

- Subsequently, the algorithm formulates an integer programming problem to compute the k-best solutions regarding the assignment of concrete services $s_{i,j}$ to each functionality f_i. To express the integer programming optimization problem in this work we adopt the concrete service utility function used in [13], which is $U(s_{j,i}) = \sum_{k=1}^{3} \frac{Q_{max}(j,k)-q_k(s_{j,i})}{Q_{max'}(k)-Q_{min'}(k)} * w_k$ where q_k(si,j) is the value of the kth QoS attribute of concrete service $s_{i,j}$ (the first QoS attribute being response time, the second cost and the third one availability), w_k being the weight assigned to the kth QoS attribute, [i.e. the maximum value of QoS attribute k among possible concrete service assignments for functionality fi], and $Q_{max'}$ [resp. $Q_{min'}$] being the overall maximum (resp. minimum) value of QoS attribute k within the service repository. Using the utility function, the computation of the best solution is expressed as the following integer programming problem: maximize the overall utility value given by $OUV_{QoS} = \sum_{i=1}^{F} \sum_{j=1}^{T(i)} U(s_{j,i}) * x_{j,i}$, where F is the number of functionalities fi requiring adaptation, and each $x_{j,i}$ is a binary variable taking the value 1 if $i_{j,j}$ is selected for delivering functionality f_i, and 0, otherwise. Since each functionality f_i is delivered in the final execution plan by exactly one concrete service, the maximization of the utility value is subject to the constraint

$$\sum_{i=1}^{T(j)} x_{j,i} = 1, 1 \leq j \leq F$$

This problem is then solved and the k-best solutions are obtained. Note that this formulation employs the sum function to rate the availability of the composite service taking into account the availability values of the constituent services, rather than the product function, as denoted in table 1.

Note that although integer programming is NP-hard, in practice, solving techniques employ a number of speed up factors namely cutting planes, presolve, branching rules, heuristics, node presolve and probing on dives [17], with each speed up factor providing a speed up ranging from 53.7% (cutting planes) to 1.1% (probing on dives);

therefore the time taken by solvers to compute the solution is much lower than the worst-case (NP-hard) complexity. The solutions are saved, together with their overall utility score, for perusal in the combination step. In order to solve the integer programming problem computing the k-best solutions, the IBM ILOG CPLEX (www.ibm.com/software/commerce/optimization/cplexoptimizer/) optimizer was used. In our implementation, we have set k=20.

4.2 The CF-based algorithm

The CF-based algorithm employed in our proposal is an adaptation of the standard GroupLens algorithm [14], modified to take into account the semantic distance of the services realizing the same functionality. For instance, rows 2 and 5 of table 2 are considered "semantically close", since they both list air transport for travel, a first class hotel for accommodation and classical music events; on the other hand rows 2 and 7 of the same table are considered "semantically distant", since all three services correspond to diverse real world counterparts (air travel vs. bus, 1st class hotel vs. 3rd class, concert vs. sports). Taking this into account, when a request arrives asking for travel via AirFrance and accommodation in GrandResort and requesting a recommendation for event attendance, the ratings in rows 2 and 7 must be taken more strongly into account than those in row 7, since the former two rows are "closer" to the one under adaptation. To accommodate this adaptation, we extend the formula of cosine similarity between two rows $\vec{X}$, $\vec{Y}$ of the UPR as follows:

$$r(\vec{X},\vec{Y}) = \frac{\sum_{k=1}^{n}(\vec{X}[k] * \vec{Y}[k] * d(\vec{X}[k],\vec{Y}[k]))}{\|\vec{X}\| * \|\vec{Y}\|}$$

We can observe in equation (2) that the standard cosine similarity metric has been extended to accommodate the semantic distance between the services that realize the same functionality in rows $\vec{X}$ and $\vec{Y}$; this is accomplished by multiplying each term of the sum in the nominator by a metric of the semantic distance between the two services, which is denoted as *d(s*$_1$*, s*$_2$*)* and is computed using the formula introduced in [15]: d(s_1,s_2) = C – lw*PathLength – NumberOfDownDirection, where *C* is a constant set to 8 [15], *lw* is the level weight for each path in subsumption tree (cf. Fig. 1), *PathLength* is the number of edges counted from functionality s_1 to functionality s_2 and *Number-OfDownDirection* is the number of edges counted in the directed path between functionality s_1 and s_2 and whose direction is towards a lower level in the subsumption tree. For more details in the computation of the semantic distance, the interested reader is referred to [15]. We further normalize this similarity metric in the range [0, 1] by dividing the result computed in the above formula by 8; this way, the multiplication by the normalized similarity metric in equation (2) reduces the correlation coefficient between the two rows by a factor proportional to the semantic distance of the services employed in these rows to realize the same functionality. For items not explicitly rated, we follow the rationale of [5] according to which usage of a service is an indication of preference, and we choose a rating equal to the 80% of the maximum rating. This is inline with the findings of [16], which asserts that dissatisfied users will provide negative feedback with a very high probability (≥89%). Rows that have not been rated at all (and therefore have a default value for all ratings) are the reason behind choosing the cosine similarity against the Pearson similarity, since

the latter disregards rows whose ratings have no variance (i.e. are all equal). Using the modified cosine similarity, the CF-based algorithm operates as follows:
1. It retrieves from the UPR all rows that contain a service implementing the functionality on which a recommendation is requested. For example, if a recommendation on event attendance is requested, only rows 1, 2, 4, 5 and 7 of table 2 will be retrieved.
2. The rows retrieved from step 1 are filtered to retain only those that fulfill the QoS criteria requested by the user.
3. The similarities between the request and each row are computed using the modified cosine similarity metric. The request is represented here as a vector vector $\vec{R}$, having a rating equal to 10 for each functionality included in the scenario and a rating equal to 0 for each functionality designated as not to be executed.
4. For each distinct service implementing the requested functionality that is included in the remaining rows, we compute its rating prediction using the standard rating prediction formula

$$p(\vec{R}[k]) = \underset{m}{mean}(\vec{R}[m]) + \frac{\sum_{\vec{N} \in raters(\vec{R}[k])} (\vec{N}[k]) * r(\vec{R},\vec{N})}{\sum_{\vec{N} \in raters(\vec{R}[k])} r(\vec{R},\vec{N})}$$

[14] (we again do not subtract the mean $\overrightarrow{N}$ from $\overrightarrow{N}[k]$, so as not to render useless the rows having only default values).
5. Finally, we retain the 20-best services, for each functionality requiring adaptation, for perusal in the combination step.
After the lists of candidates for each individual service that is subject to adaptation have been computed, the algorithm selects the top-20 execution plans with respect to their CF-score. Given an execution plan containing services (s1,i, …, sN,k) with the similarity scores of the services computed in step 5 being (CFS(s1,i), …, CFS(sN,,k)), then the CF-score of the execution plan is equal to CFS(s1,i)+…+CFS(sN,k). Computing the top-20 execution plans is modeled as an integer programming optimization problem, formulated in a similar fashion to the one described in section 4.1. Full details on the formulation of the integer programming optimization problem are given in [12]. The CF module has been implemented using Apache Mahout (https://mahout.apache.org/), by subclassing the UncenteredCosineSimilarity class and reimplementing in the subclass the UserSimilarity method, to accommodate the semantic similarity metric described above.

4.3 The combination step

The combination step synthesizes the results given by individual algorithms to produce a single result. Recall from the previous two subsections that each algorithm produces a set of candidate execution plans, with each execution plan being tagged with the relevant normalized score (QoS-score or CF-score). In order to combine the scores, we use the CombMNZ metasearch algorithm, since it has been found to have the best performance [3] [the CombMNZ rating of a solution is computed by multiplying the sum of the individual scores by the number of non-zero scores, i.e. where mi is the number of algorithms giving non-zero rating to item *i* and $r_j(i)$ is the rating given by algorithm *j* to item *i*]. After computing the CombMNZ metasearch for all candidate execution plans, the combination step selects the execution plan with the highest score, which will be used to drive the adaptation process.

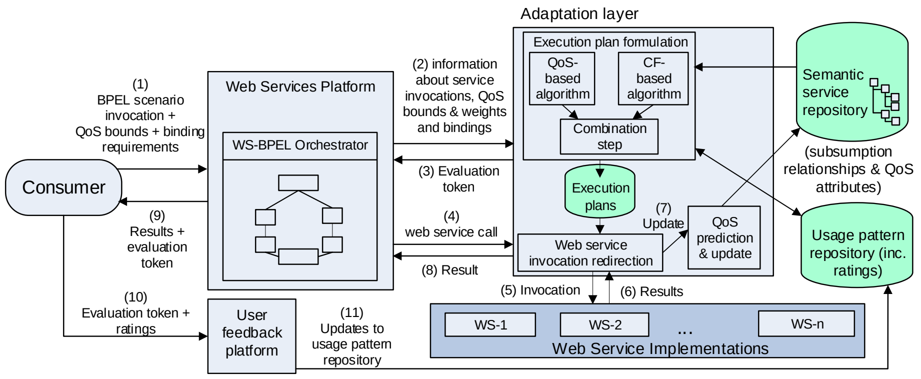
Adaptation layer
Execution plan formulation
QoS-based algorithm
CF-based algorithm
Combination step
Execution plans
Web service invocation redirection
(7) Update
QoS prediction & update
Semantic service repository
(subsumption relationships & QoS attributes)
Usage pattern repository (inc. ratings)
Web Services Platform
WS-BPEL Orchestrator
Consumer
(1) BPEL scenario invocation + QoS bounds + binding requirements
(2) information about service invocations, QoS bounds & weights and bindings
(3) Evaluation token
(4) web service call
(8) Result
(9) Results + evaluation token
(10) Evaluation token + ratings
User feedback platform
(11) Updates to usage pattern repository
(5) Invocation
(6) Results
WS-1
WS-2
...
WS-n
Web Service Implementations

Fig. 2. The execution adaptation architecture

4.4 The execution adaptation architecture

The execution adaptation architecture, illustrated in Fig. 2, follows the middleware-based approach, with an adaptation layer intercepting web service invocations and appropriately directing them to the services chosen by the adaptation algorithm. As shown in Fig. 2, the BPEL scenario execution initially passes to the adaptation layer the information regarding service invocations that will be performed, QoS bounds and weights as well as specific service bindings. When the adaptation layer receives this information, it applies the adaptation algorithm to formulate the execution plan for the particular scenario execution (i.e. decide the actual services that will be invoked to deliver each functionality) and stores the execution plan for later perusal. Subsequently, when a web service invocation is intercepted by the adaptation layer, the respective execution plan is retrieved from the execution plan storage, the web service decided to deliver the specific functionality is extracted and the invocation is routed to that service. Note that steps (4)-(8) depicted in Fig. 2 are repeated multiple times within each BPEL scenario execution, once per web service invocation performed. When the invocation to a service implementation has concluded, the data regarding the service's response time and availability are passed to the QoS prediction and update module, which computes the predicted values for the respective QoS parameters and updates the corresponding elements in the semantic service repository. Additionally, the BPEL scenario returns at the end of its execution, along with the result, an evaluation token, which the consumer may use to enter the ratings for the services s/he has used in the context of the BPEL scenario execution. The evaluation token is returned in the response headers, to retain the response payload schema intact. To accommodate this additional functionality (passing the necessary information to the adaptation layer and returning the evaluation token), the BPEL scenario is preprocessed as described in [5] before being deployed to the web services platform, with the preprocessing step injecting the necessary invocations to the adaptation layer into the scenario, and the result of the preprocessing step is then deployed and made available for invocations.

5 CONCLUSIONS AND FUTURE WORK

In this thesis we have presented a framework for adapting the execution of BPEL scenarios, taking into account data from the monitoring of the QoS offered by the services, as well as user ratings. To perform the adaptation, we follow the metasearch paradigm, by combining two candidate execution plan ranking algorithms. The first one examines the execution plan QoS aspects only, while the second is based on CF techniques. The framework provides means for monitoring the QoS parameters of the services and adjusting accordingly the values of the services' QoS attributes, as well as accepting user ratings for the services they have used, which are taken into account by the CF-based algorithm. The proposed framework is complemented with an execution architecture for enacting the adaptation, which adopts the middleware approach, with an adaptation layer intervening between the BPEL execution platform and the web services and arranging for redirecting service invocations to the services selected by the adaptation algorithm. The proposed framework has been experimentally validated regarding (i) its performance, (ii) the quality of execution plans generated and (iii) the effectiveness of the QoS monitoring and estimation mechanisms. The proposed approach has been also found to be scalable, exhibiting a linear increase in the imposed overhead.

Our future work will focus gathering statistical information from prior scenario executions and using it as input to the adaptation process. This information will quantify aspects regarding the behavior of control constructs in the scenario. We also plan to examine how the algorithm can be extended to consider different adaptation strategies.

6 REFERENCES

[1] OASIS WSBPEL TC. WS-BPEL 2.0. http://docs.oasisopen.
org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[2] Kareliotis, C., Vassilakis, C., Rouvas, S., Georgiadis, P. QoS-Driven Adaptation of BPEL Scenario Execution. In: *Proceedings of ICWS 2009*, 271-278, 2009.

[3] Montague, M., Aslam, J.A. Relevance score normalization for metasearch. In: *Proceedings of CIKM 2001*, 427-433, 2001.

[4] Margaris, D., Vassilakis, C., Georgiadis, P. An integrated framework for QoS based adaptation and exception resolution in WS-BPEL scenarios. In *Proceedings of the 28th ACM SAC*, Coimbra, Portugal, 1900-1906, 2013.

[5] Margaris, D., Vassilakis, C., Georgiadis, P.: Adapting WSBPEL scenario execution using collaborative filtering techniques. In: *Proceedings of the IEEE 7th RCIS Conference*, Paris, France, 2013.

[6] Paolucci, M., Kawamura, T., Payne, T., Sycara, T. Semantic Matching of Web Services Capabilities. In: *Proceedings of the 2002 International Semantic Web Conference*, 333-347, 2002.

[7] Shao, L., Guo, Y., Chen, X., He, Y. Pattern-Discovery-Based Response Time Prediction. In: *Advances in Automation and Robotics*, vol. 2, LNEE, vol. 123, 355-362, 2012.
[8] Duan, Y., Huang, Y. Research on availability prediction model of web service. In: *Proceedings of the 2011 International Conference on Computer Science and Service System*, 1590–1594, 2011.
[9] O'Sullivan, J., Edmond, D., Ter Hofstede, A. What is a Service?: Towards Accurate Description of Non-Functional Properties. *Distributed and Parallel Databases*, 12, 2002.
[10] Canfora, G., Di Penta, M., Esposito, R., Villani, M.L. An
Approach for QoS-aware Service Composition based on Genetic Algorithms. In: *Proceedings of the 2005 Conference on genetic and evolutionary computation*, 1069-1075, 2005.
[11] Yu, J., Sheng, Q., Han, J., Wu, Y., Liu, C. A semantically enhanced service repository for user-centric service discovery and management. *Data & Knowledge Engineering*, 72, 202- 218, Feb. 2012.
[12] Margaris, D., Vassilakis, C., Georgiadis, P. Combining Quality of Service-based and Collaborative filtering-based techniques for BPEL scenario execution adaptation. University of Peloponnese SDBS Technical report TR-14002, 2014, available at http://sdbs.dit.uop.gr/?q=TR-14002
[13] Alrifai, M., Risse, T. Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition. In: Proceedings of the 18th international conference on World Wide Web, 881-890, 2009.
[14] Saric, A., Hadzikadic, M., Wilson, D Alternative Formulas for Rating Prediction Using Collaborative Filtering. In: Proceedings of the 18th International Symposium on Foundations of Intelligent Systems, 301-310, 2009.
[15] Bramantoro, A., Krishnaswamy, S., Indrawan, M. A semantic distance measure for matching web services. In: Proceedings of the 2005 International Conference on Web Information Systems Engineering,. 217-226, 2005.
[16] Chelminski, P., Coulter, R. An examination of consumer advocacy and complaining behavior in the context of service failure. Journal of services marketing, 25, 5, 361–370, 2011.
[17] Bixby R.E., Fenelon M., Gu Z., Rothberg E., Wunderling R. Mixed integer programming: A progress report. Chapter in Martin Grötschel (ed.), The sharpest cut: The impact of Manfred Padberg and his work, MPS-SIAM Series on Optimization, Vol. 4, 2004
[18] D. Margaris, C. Vassilakis, P. Georgiadis, "An integrated framework for adapting WS-BPEL scenario execution using QoS and collaborative filtering techniques", Science of Computer Programming, Volume 98, Part 4, 1 February 2015, Pages 707–34, http://www.sciencedirect.com/science/article/pii/S0167642314004778
[19] D. Margaris, C. Vassilakis, P. Georgiadis, "A hybrid framework for WS-BPEL scenario execution adaptation, using monitoring and feedback data", to appear in the 30th ACM Symposium on Applied Computing, Salamanca, Spain, 2015.