Effective Automatic Dynamic Semantic Web Service Composition

Peter Bartalos[*]
Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
bartalos@fiit.stuba.sk

Abstract

Web services are a popular technology used when diverse software system integration is in demand. The ability to make some functionality available through the Web has inspiring consequences. One of the intensively researched areas is the study how Web services can be used to dynamically create a functionality, based on the actual requirements. The basic idea is that multiple Web services can be combined together to form a composite service supplying more complex needs. The composition is realized automatically, on the fly, based on the actual goal. To facilitate this kind of Web service utilization, additional metadata depicting the functionality of single services is required. These metadata are provided in a form of semantic annotations. Our work deals with selected sub-problems of the automatic dynamic semantic Web service composition. The sub-problems include the proper description of the behavior of Web services, management of the changes in the service environment, and handling multiple composition requests arriving continuously.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Service-oriented architecture (SOA); D.2.13 [**Reusable Software**]: Reusable libraries; H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness); H.3.5 [**On-line Information Services**]: Web-based services; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Semantic networks

[*]Recommended by thesis supervisor:
Prof. Mária Bieliková
Defended at Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava on February 2, 2011.

Bartalos, P. Effective Automatic Dynamic Semantic Web Service Composition. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 3, No. 1 (2011) 61-72

Keywords

Web services, service composition, semantic web services, QoS, pre-/post-conditions

1. Introduction

Web services are showing to be important in the IT world. This concerns both the practice and research. In practice, the Web service technology allows to make diverse software systems interoperable, independently on the platform and place they operate on. Thus, Web services play important role in a development of distributed applications. The benefit is also a reuse of loosely coupled, pervasive software components. In the research area, Web services are a concept, studied from several view points, to develop approaches providing a functional content on the Web in a sophisticated manner. This leads to a more interactive Web, allowing finding and using high quality services. The aim is to deliver huge number of Web services providing a comprehensive functionality, covering various areas, from E-Government to entertainment.

The idea of the Web providing valuable functionality in a form of services, requires a development of various methods dealing with a wide range of problems. These relate to the modeling and description of Web services, quality of services, security and trust, discovery and composition, transactional behavior, etc. These issues bring forward several research challenges. Our work focuses on a Web services composition, aiming to arrange multiple services in a meaningful manner, to supply complex needs.

To be able to supply varying user goals, arriving continuously, the composition must be dynamic. This means that it is performed on the fly, based on the actual user goal. Due this, it is desired to realize it in automatic manner, i.e. by an intelligent software. This requires that the Web services are present together with a machine processable description. The knowledge available in these descriptions must unambiguously define what functionality the service provides, what are the required inputs and produced outputs. These aspects of the Web services are studied in the context of Semantic Web services.

The actual arrangement of multiple services, based on a given goal, is a high computation demanding process. Different methods, having their base in artificial planning approaches, are usually applied here. Considering the processing of the semantic metadata of services, optimization of the composition according to the QoS (Quality of Service), the problem tends to be NP-hard. The number of Web services, which are considered to be composed, is expected to be high and rising. Thus, the performance and scalability issues have high importance in the area too.

This paper presents an extended summary of dissertation. We provide an overview of the field of Web services composition. We focus on issues related to semantics, artificial intelligence planning, and quality of services. This is followed by the objectives of the dissertation. Then, we overview the basic concepts of our methods, used to achieve the introduced objectives. In this context we presents also a summary of the evaluation of our approach. After this, we present and discuss the contributions of the thesis. Finally, we conclude our work. The paper includes also a list of selected papers of the author.

2. State of the Art

The Web was primarily designed for use by humans. The need of automatic processing of Web resources by machines brought forward a need for machine processable representations of semantically rich information: a vision at the heart of the *Semantic Web* [12]. The additional metadata, depicting the semantics, concerns both the content information and services available on the Web. The added metadata are used to enhance the processing of Web resources (documents, services). The base of the semantic enrichment is to bind the elements of Web resources to domain terms defined within an ontology. In the case of Web services, the semantic annotation primarily focuses on the inputs and outputs. Web services which are provided together with semantic annotations are called Semantic Web services [14, 15, 27, 31, 41]. Several Web service annotation languages have been proposed to be used for representation of the semantic metadata. The most known are: *OWL-S*[1,2], *WSML*[3,4], *USDL*, and *WSDL-S*[5] [29, 35, 25, 33].

During composition, the semantics helps to discover and arrange services in a meaningful manner, according to the given goal [17, 40, 10, 42]. Several works propose that, to correctly understand the behavior of a Web service, it is not enough to have the semantic information concerning the I/O. A more natural way to express some behavior is by a *cause-effect* paradigm. Due this, Web services should have defined the pre-/post-conditions to better express what they do. This kind of service modelling helps to improve the way how the user expresses his goal, service discovery, and composition [10, 42]. On the other side, considering the pre-/post-conditions of the services makes the composition problem more challenging.

The pre-/post-condition aware service composition is addressed in [24, 23, 39, 28, 21]. Slight differences can be observed between these approaches concerning the definition of a condition. Some approaches allow to define only a conjunction of predicates. Others allow to use also the logical disjunction to depict some alternative conditions. The expressivity of the conditions is important considering the complexity of the approach. The processing of the pre-/post-conditions presents a challenge also from the performance point of view.

The approaches to Web service composition assume that the semantic metadata of services is correct. Unfortunately, it is showing that the creation and maintenance of ontologies and the semantic metadata of Web resources still deals with problems. Moreover, even if we have correct ontologies, it is not assured that any intelligent software will be able to understand the semantics the same way. These problems are studied as a field *ontology matching*, or *alignment* [16]. Several methods, estimating the semantic relation of two concepts, associated with web service elements, had been developed [45, 43, 9, 44]. These help to make the processing of the semantics more precise.

The core problem of automatic web service composition, which is to design a prescription based on which the proper services are invoked in the right order, is equivalent to the *artificial intelligence planning* problem. This means that the basic idea behind different artificial intelligence planning methods could be applied to service composition. The most important are *State-space based planning*, *Graph based planning*, *Backward chaining*, *Planning based on logical programming*, *Rule-based planning*, and *Hierarchical task network planning.* The problem of finding a suitable composite service can be solved by transforming it to a particular planning problem [34, 32]. If we consider additional requirements, specific for the web service composition, the planning approaches must be adapted and enhanced.

The artificial intelligence planning approaches are the base for most of the existing Web service composition methods. The most used from them are different refinements of state-space search and planning graph [19, 20, 26, 36, 38, 11, 21]. The approaches based on these methods usually perform the QoS driven composition. Their benefit is that we can enhance them to meet the particular needs of the Web service composition. For example, the search can be enhanced to include optimization according to a desired property. These approaches had also shown to be the best option when performance is an issue.

Several Web service composition methods base also on logical programming [24, 23, 39, 28]. These usually deal with the pre-/post-condition aware service composition. They benefit from the use of existing components such as the reasoners. The attention here is given to the translation of the problem into a logic programming domain and the interpretation of the results. Although that the logical programming based approaches natively support the processing of the pre-/post-conditions, they are harder to enhance to deal for example with the QoS. We have also less control over the performance of the approach, since it depends on the used reasoner. These issues are easier to deal with, when state-space planning or graph plan based approach is used. On the other side, these do not give us support to deal with the pre-/post-conditions.

A special and often used method, to compose services, is hierarchical task network planning. It is suitable in domains where the hierarchical decomposition of the problem can be applied. Due the diversity of the Web service composition problem (e.g. if it is QoS aware, or if it considers the pre-/post-condition), there is no absolute win-

[1] `http://www.daml.org/services/owl-s/1.0/owl-s.html`
[2] `http://www.w3.org/Submission/OWL-S/`
[3] `http://www.wsmo.org/wsml/`
[4] `http://www.w3.org/Submission/WSML/`
[5] `http://www.w3.org/Submission/WSDL-S/`

ner considering different planning approaches. Instead of putting all the approaches into a position of competitors, we should see them as complementing. Which approach should be used depends on the concrete situation. There also exist approaches combining different methods to exploit the benefits of each of them [28, 21]. These are showing to be useful when various aspects of the Web service composition are considered at the same time.

The optimization of the service composition, according to the QoS, is in attention of several approaches. Their aim is to, beside the functional properties of Web services, consider also the non-functional attributes. Based on these, from all the possible solutions, we select that one, which has to best total asset considering the QoS. The QoS include various attributes, which are represented numerically, or can be transformed into a numerical representative. Usually, the technical attributes such as response time, throughput, availability are considered. Each attribute has defined aggregation rules. Based on these, we calculate a global QoS value for the whole composition. During this we consider the QoS values of single services used in the composition [48, 47, 2, 26, 36].

To get an overall quality measure representing the quality in one number, we calculate a uniform quality representative based on a utility function. It aggregates different QoS attributes, whose values might be of different units and ranges. The resulting value is used to rank the compositions. The calculation of the utility function is usually based on *Multiple Attribute Decision Making* method, having its origin outside of the QoS driven service composition [46]. This method is based on scaling and weighting [48, 2, 38]. The scaling allows a uniform measurement of multi-dimensional attributes, independently on their units and ranges. The weighting allows to express preferences over different quality attributes.

3. Goals

The aim of this thesis is to deal with the problem of automatic dynamic semantic Web service composition. The main objective is to deal with problems related to the following issues:

- *Functional aspects of Web services.* Proper representation of the functional aspects of Web services is crucial for automatic Web service composition. The existing approaches exploit additional meta-data depicting the semantics of the I/O parameters to describe the service behavior. This approach shows to be insufficient. The proposed solutions are oriented to express the pre-/post-conditions of Web services. Our aim is to clearly define the pre-/post-conditions and show that it is a feasible way to compose services while considering their pre-/post-conditions.
- *QoS optimization.* Beside the functional requirements, the user is usually interested also in non-functional properties of Web services. Hence, the QoS optimization during service composition is important. We deal with a service composition aware of the QoS and capable to find the best solution considering them.
- *Changes in the service environment.* The Web service environment is frequently changing in time. New services are deployed, some of them are removed. The changes relate also to the QoS attributes, whose values might evolve in time. Our goal is to develop a composition approach capable to react to these changes and thus providing a solution reflecting the actual situation in the service environment.
- *Effectiveness.* As the Web in general grows, also the set of Web services which are available in repositories is rising. We deal with the problem of performance and scalability of the composition process considering large number of services to be searched and composed.
- *Composition system.* Our aim is not only to develop a composition method, but also to design a composition system realizing it.
- *Continuing user query arrival.* In real scenarios, the composition system must dynamically compose services based on the actual user goal. The composition queries may arrive from multiple users. We study the behavior of the composition system due continuing user query arrival, while reacting also to possible frequent changes in the service environment.

4. Fast and Scalable Web Service Composition

The basic steps of the composition process are:

- finding the *initial services*,
- finding the *final services*,
- design of the *control-/data-flow* of the composite service along with QoS optimization.

The initial services are those having all inputs provided by the user. Thus, they can be potentially executed which means that if they are helpful to satisfy the user goal, we can include them in the composite service.

By finding the final services, we determine which services directly produce the user goal, i.e which services have outputs and post-conditions required in the user goal. If these services have provided all inputs by the user, the resulting composite service includes only the final services. Usually, this is not fulfilled. In general, we have to deal with construction of a composite service including the initial, final, and intermediate services. The initial services produce outputs and conditions required by the intermediate services. The intermediate services include multiple interconnected services producing the outputs and conditions required by the final services.

The overall composite service prescribes the order in which the individual services must be executed and how the outputs are passed to services requiring them as inputs. Usually, multiple composite services can be found which taking the provided inputs produce required results. To find a solution satisfying the user needs the best, we consider the QoS characteristics of the individual services and select a solution with the optimal aggregated QoS value of the given attribute.

The finding of the initial, final, and intermediate services requires to evaluate the compatibility of two i) concepts, and ii) conditions. The semantic types are considered between pairs consisting from inputs provided in the user goal and service inputs, service outputs and inputs, or service outputs and outputs required in the user goal. Analogically to semantic types, we need to consider also the pre-/post-/goal-conditions.

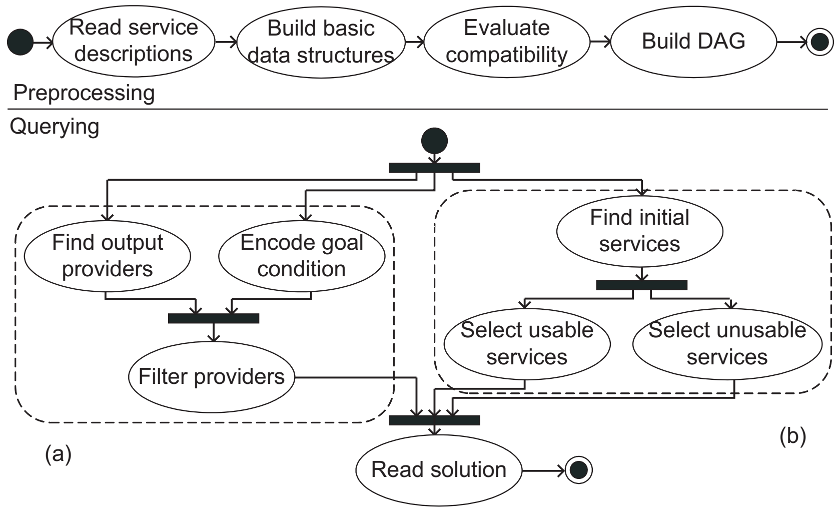

Figure 1: Overview of the composition process.

The computations required to compose a service satisfying the user goal might be significantly time consuming. There are several issues affecting the computation demands:

- number of services in the registry,
- number of service inputs, outputs[6],
- complexity of the pre-/post-/goal-conditions,
- the interrelations between services resulting from their compatibility,
- the fact if we look for the optimal solution from QoS point of view, or not.

In general, several operations required during service composition have a complexity exponentially rising based on certain parameters. Some examples are the design of the control-/data-flow of the composite service, or evaluation of the compatibility between two conditions. To be able to compose services in acceptable time, sophisticated methods overcoming the high computation demands must be developed. The main issues making our approach good performing are preprocessing, effective data structures and algorithms.

The preprocessing is crucial to quickly respond to composition queries. In our approach, we perform preprocessing before responding to user queries, see top of Fig. 1. During it, we analyze the actual service set and build data structures which are used to quickly answer the query. All the important computations, which can be done without the knowledge of the user goal, are realized during preprocessing.

The most important is that we evaluate which services are compatible and can be chained. Based on the compatibility check a directed acyclic graph (DAG) of services is build. Each service is represented by a node. If two services can be chained, there is a directed edge from the ancestor to the successor service. During chaining evaluation, we consider also possible data adaptation operations, which are performed if the outputs must be adjusted to fit the inputs. These operations include composition, decomposition, listing, and list arrangement.

[6]services might have multiple outputs in general

The still remaining task is the finding of the initial, final services, and the design of the data-/control-flow, see bottom of Fig. 1. These can be completely realized only when the user goal and the provided inputs are known. Fig. 1-a) concerns the compatibility evaluation. When looking for the final services, we need to find services producing the required data and having a post-condition implicating the goal condition, i.e. have results compatible with a user goal. Fig. 1-b) relates to the creation of the control-/data-flow and the optimization according to the QoS. During all these tasks, the effective data structures, built during preprocessing, are used to realize these tasks quickly. During the operation of the composition system, the update of the data structures could be required to handle the dynamic changes in the service environment.

4.1 Concepts and Conditions Compatibility Evaluation

As mentioned before, during the service composition we have to deal with the compatibility evaluation between concepts and conditions. Considering two concepts and conditions, the question is if the first concept subsumes the second, and if the first condition implicates the second. The compatibility check is required during preprocessing when the directed acyclic graph of services is built and during querying when we look for the initial, and final services. More precisely, the problem is reformulated to finding of services having outputs and post-conditions, or inputs and pre-conditions which are compatible with a given concept and condition. In other words, we have a pair of a given concept and condition, and look for a set of services being compatible with them.

In the case of finding the initial services, the aim is to find services having inputs associated with concepts subsuming the concepts associated with the inputs provided by the user. When looking for the final or ancestor services (during DAG building), the given concepts are those associated with outputs required by the user or the inputs of the successor services. The aim is to find services with outputs associated with concepts subsuming the given concept. Considering the conditions, the found services must also have a post-condition implicating the given condition. The given condition is the goal-condition, or the pre-condition of the successor service.

The finding of services having inputs, or outputs associated with the given concept is much simpler than evaluating the implication between two conditions. Since the set of services which are considered during the composition is known already before we process a composition query, we can create effective data structures during preprocessing which makes the finding of the services fast. We use two hash tables with the same structure. The first is used to find services having outputs associated with the given concept and the second is for services having inputs associated with the given concept. In both of them, the keys are all existing concepts and the values are lists of services. In the first case, the list includes services having outputs associated with the concept used as a key. In the second case, the list includes services having inputs associated with the concept used as a key.

By using the hash tables, created during preprocessing, we achieved that the finding of the services producing, or requiring data associated with the given concept is done in constant time, i.e. the complexity of this procedure is $O(1)$. The finding requires only to get the item in the hash table stored under the key equal to the given concept. Hence, the finding of the final services, neglecting the post-/goal-conditions, is done in $\Theta(|O'|)$ time at all ($|O'|$ is the number of required outputs, defined in the user goal). Analogically, the finding of the initial services requires $\Theta(|I'|)$ time ($|I'|$ is the number of provided inputs, defined in the user goal).

If it is required to check also the condition implication, it is always performed after the concept compatibility check. Instead of finding services producing desired post-condition in the whole service registry, we only check whether the services producing data associated with the required concepts, satisfy also the post-condition restriction, i.e. we filter the found services based on post-conditions. This way we avoid wasting time by evaluating condition implication in those cases, when despite the result, we cannot use the service because of incompatible data from semantic point of view.

We have developed two approaches to evaluate the condition implication. The first bases on relational databases (RDB) and the second uses condition encoding. The RDB based approach stores the data structures in a relational database. To find services having a post-condition, implicating a defined condition, a query is performed. The results of the query are processed programmatically, to evaluate the mapping between the variables appearing in the two conditions as the predicate arguments. After this step, all the fitting services have been found.

Our second approach to condition implication evaluation is based on encoding some specific properties of the conditions. We precalculate several characteristics of the conjunctions and their elements. The selection of characteristics is based on our analysis of what can be relatively easy to calculate and strongly specific for a conjunction and variables appearing in it. The characteristics are encoded as a unique number having such a property that, any two subjects of encoding having the same code are equivalent based on the properties, which are considered in that encoding. Based on the encoding, we can quickly decide if two conditions are compatible, or not.

The experimental results concerning compatibility evaluation are presented in Tab. 1–3. The complexity of the conditions is depicted by the average sizes of the disjunctions (Str) and conjunctions (str), considering the conditions transformed into disjunctive normal form. The time is presented in milliseconds.

Tab. 1 presents the compatibility evaluation times achieved using the RDB based approach. Tab. 2 present the results of the encoding based approach. In Tab. 3 we see how much times is the encoding based approach faster than the RDB based approach. In any test case, the encoding based approach showed to be more effective. However, as the complexity of the conditions rises, the difference is smaller. Hence, the RDB based approach scales better than the encoding based approach.

Table 1: RDB approach

$\|str\|$ \ $\|Str\|$	12	20	30	37
5	72	116	150	185
10	120	167	251	277
15	220	255	288	309
20	345	366	382	569

Table 2: Encoding approach

$\|str\|$ \ $\|Str\|$	12	20	30	37
5	17	24	37	49
10	23	45	63	82
15	42	75	100	141
20	64	111	159	213

Table 3: Comparison

$\|str\|$ \ $\|Str\|$	12	20	30	37
5	4.24	4.83	4.05	3.78
10	5.22	3.71	3.98	3.38
15	5.24	3.40	2.88	2.19
20	5.39	3.30	2.40	2.67

4.2 Control- and Data-flow Design

In general, a composite service satisfying a user goal consists of several services, which must be executed in a specific order. This section explains the process of finding services which are used in the composition to produce data for the final services, if these are not provided in the user query. In general, this set in not empty. The empty set refers to a situation that all the final services have inputs provided in the user query. Usually, this set consists of several interconnected services.

Our approach is based on two processes. The first selects services which have provided inputs and thus can be used in the composition. The second selects services which cannot be used because they do not have provided all inputs. The inputs are provided in the user query, or as an output of another usable service. The second process is not necessary to find a composition. It is used only to faster the selection of the usable services, which is a necessary process.

Table 4: Experimental results.

Services	**Composition time (msec)**			**Number of code line crosses**					
	Par	**Seq**	**NoUnusab**	**Par A**	**Seq A**	**NoUnusab A**	**Par B**	**Seq B**	**NoUnusab B**
10 000	6	7	97	991	976	30 767	552	149	5 079
20 000	11	19	336	1 728	1 611	53 686	831	263	9 249
30 000	42	49	718	3 041	3 018	72 825	539	319	12 325
40 000	29	44	932	1 144	1 136	52 368	606	204	8 438
50 000	22	49	1 022	1 674	1 661	55 542	376	248	12 023
60 000	60	94	1 454	2 613	2 581	62 142	1 361	199	11 645
70 000	82	106	2 070	1 577	1 413	76 288	751	254	12 713
80 000	76	75	2 806	2 194	2 174	76 390	602	290	11 230
90 000	173	222	2 613	3 299	3 262	50 183	471	329	11 025
100 000	121	179	3 009	2 711	2 667	75 202	895	256	14 589

To select usable services, forward chaining is realized. In its base, it is a graph planning approach, starting with the initial services. During it, we also select the best provider for each input of all services from the QoS point of view. After, the solution is read backward, starting with the final services. The backward reading continues through each input, which is not provided in the user query, over all services already in the read composition. Considering QoS, we select those providers of the inputs having the best aggregated value of a particular QoS attribute. We do not deal with calculation of the uniform quality representative of the composite service, providing a unified view to the quality of the composition.

To speed up the *select usable* services process, we propose service space restriction. Its aim is to select services which are unusable. A service is unusable, if at least one of its inputs is not provided in the user query, neither as an output of the ancestor services. The process lies on identification of such services, for which there is at least one input not provided by any available service, i.e. the only case when it is usable is when the respective input is provided in the user query. These services are identified during preprocessing. We call them *user data dependent services*.

Our experiments show, that our approach is capable to design the control-/data-flow in acceptable time, even if the number of considered services rises up to 100 000, see the experimental results presented in Tab. 4. This is possible due the effective data structures and preprocessing. The experiments also show, that the service space restriction dramatically improves the performance. We have been experimenting with different configurations of the composition process concerning the service space restriction. First, the restriction is performed in parallel with selection of the usable services (denoted as Par). Second, the usable services are selected after the restriction is finished (denoted as *Seq*). Third, the service space restriction is not applied (denoted as $NoUnusab$).

To clarify the reason of the improvement, caused by service space restriction, we measured also how much times did the execution of the *select usable* services process cross two parts of the algorithm, denoted as A and B, which are in the two most inner loops of it. The decrease of the crosses explains the improvement of the composition time. It is caused by the fact that we do not waste time by processing services, which are not usable in the composition. The Par configuration presents an improvement from 15 to 46 times in terms of composition time and adequate improvement in terms of the number of crossing parts A, B. From all the configurations, it performed the best. The Seq configuration performs slightly slower. According to a case when no service space restriction is applied, both Par and Seq configuration present dramatic improvement.

4.3 Dynamic Changes in The Service Environment

As it is true for the Web in general, also Web services change in time. New services are deployed, some of them are removed. Moreover, the non-functional properties of services may change frequently. The aim is to find a composite service with the best global QoS characteristic. The composition system must flexibly react to these changes. The solution should reflect the current situation in the service environment. If the changes are not managed, it may happen that the designed composition does not use the right services, it includes services which are already not executable, or are not optimal from QoS point of view. Hence, we do not achieve the satisfaction of the user goal, based on which the composition is realized. On the other side, the dynamic changes of services require updating the data structures of the composition system, before we start new composition. Thus, the dynamic changes affect the composition time. To avoid too much delay, the updates must be realized quickly.

There are three types of changes we consider in the service environment:

- adding a service,
- removing a service,
- change of the QoS of a service.

The change of the service QoS characteristics are managed in constant time, independently on the number of services. It requires only changing the values of the QoS attributes in the data structure, representing the corresponding service. The updates required because some service is made (un)available are divided into two cases.

If a new service is made available, we have to analyze its I/O and pre-/post-conditions, to be able to connect it into a DAG of services data structure. If some service is removed, we have to disconnect it from DAG. Since existing services might go down temporally, it is not desired to disconnect and re-connect it all the time. Instead of doing this, we can set a flag of a service, denoting its availability, which is considered during the composition. Thus, we know if it can be used, or not. Since the setting of the flag is done in constant time, this operation does not cause performance problems.

Table 5: Operation times (in msecs).

Web services	Add	Remove	Reinitialization	Composition
10 000	0.84	1.68	1.86	4.95
20 000	0.92	2.84	4.46	14.8
30 000	1.02	4.82	10.2	46.3
40 000	1.53	7.88	13.8	35.9
50 000	1.13	5.81	19.6	37.3
60 000	2.12	10.3	25.6	93.6
70 000	1.39	9.11	27.1	88.0
80 000	1.48	13.3	29.5	64.3
90 000	1.86	9.46	30.0	271.8
100 000	1.89	12.0	51.1	152.4

In Tab. 5 we see our experimental results concerning the changes in the service environment. It presents the times required to add a new service, and permanently remove a service, compose services, and reinitialize the system required after each composition. As we can see, the time required to add/remove a service is significantly lower than the composition time. This is necessary in practical scenarios, where the services are changing dynamically. Our approach showed to be able to handle this requirement.

4.4 Continuing Query Arrival

To fulfill the requirement that the composition system must be able to process multiple continuously arriving requests, we built it as a queuing system with different type of requests [1]. The composition and the different type of update requests are collected in separate queues. It works as depicted in Fig. 2. When the system starts, it initializes its data structures based on currently available services. After, it is waiting for update requests, or user queries (1). These are collected in queues and processed according the *first in first out* rule. The updates are managed with higher, *non-preemptive* priority, i.e. the composition is not interrupted if an update request arrives. If an update request arrives (2), i.e. new service is available, some service becomes unavailable, or the QoS characteristics of some service had changed, we update the affected data structures. When all the changes are managed (3), the system is ready to process a new user request. If a new query has already been received, the system processes it immediately (4). Otherwise, it goes to waiting state (5). Here, it again waits for an update request, or composition query (6). The composition is followed by the reinitialization (7). After, we manage update requests, if some had arrived (8). If not, we process a new user query for composition (9), or go to the waiting state (10).

As just introduced, the base of the composition system is collecting the requests and their processing one after the other. This means that, if considering a single instance of the composition system, it happens that some requests are not processed immediately. This causes that the overall processing time of a request, is longer than the pure time spent by composition. The overall processing time in this context is called the sojourn time. The additional delay is dependent on the actual load of the system. While the interarrival time between the individual requests is long enough, the system is stable and the additional delay in processing time is low. In this case also the queue size is low. Note that there is a linear dependency between the sojourn time and the queue size. If the requests arrive too frequently, the system is not capable to process them quickly enough (even if it works on 100%) and becomes overloaded. In this case the system is unstable. Both the sojourn time and the queue size rise, in time, eventually to infinity.

Fig. 3 presents the experimental results concerning the processing of continuing composition requests. It depicts the sojourn time according to the mean interarrival times of the requests. We had been experimenting with two distributions of the interarrival times: exponential and uniform. In real scenarios we assume that the requests are continuous and independent, thus occur according to a Poisson process. In this case, the interarrival times follow exponential distribution.

The experiments show, that the composition is stable, if the mean interarrival time of the requests is more than a double of the pure composition time. In this case, the additional delay causes that the sojourn time is no more than a twice of the composition time. This means that the system is capable of long-term, stable performance and processes the requests still in acceptable time. If the mean interarrival times between the requests is less than a double of the composition time, the system tends to be exhausted and unstable.

During our experimentation with continuing arrival of requests, we tested also how is the performance of the composition system affected by the dynamic changes in the service environment. Beside the composition requests, we generated also service changes, i.e. adding and removal of services. Fig. 4 shows, that our system is capable to process the changes effectively, without causing significant delay in the sojourn time of composition requests. This is true also in the case that the changes are very frequent, e.g. each 10 msec. This proves that our system handles the changes in the service environment effectively.

4.5 Composition System Architecture

Based on the defined requirements, we designed a software architecture of the composition system as presented in Fig. 5. The system is divided into two main subsystems. The first includes components responsible for the preprocessing (bootstrap) phase. The second is responsible for the user querying phase. The functionality of the system is provided via the following interfaces:

1. initialize(wsdlURLs, owlURLs, wslaURLs): initializes the composition system based on the initial web service set.
2. startQuery(wsdlQuery, callbackURL): invokes the composition process for the defined goal.
3. stopComposition(): stops the current composition process (could be used in the case when the system

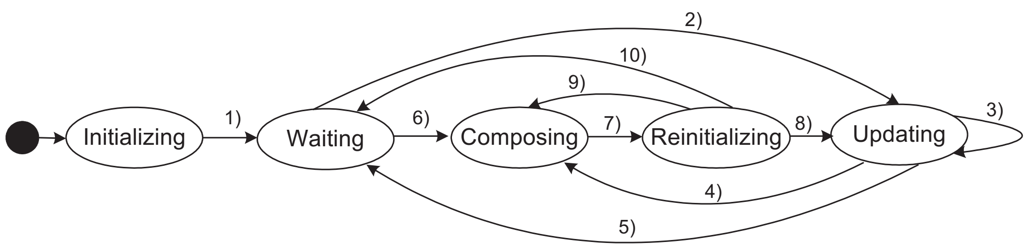

Figure 2: Composition system life cycle.

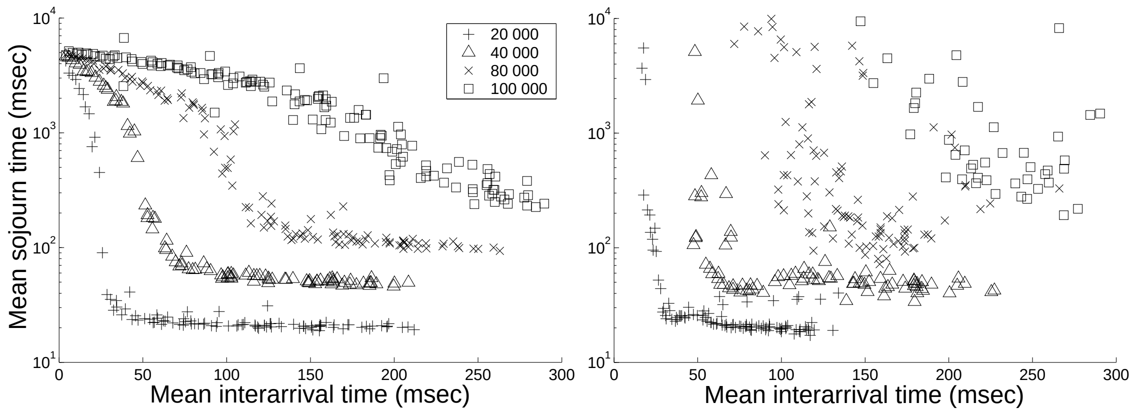

Figure 3: Sojourn time: uniform distribution at left, exponential distribution at right.

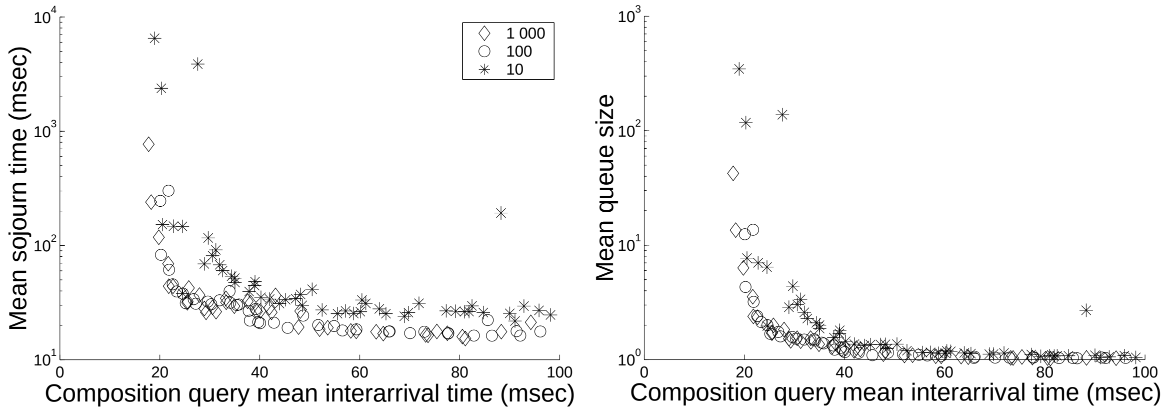

Figure 4: Effect of dynamic changes in the service environment.

is unable to realize the composition in acceptable time).

4. updateQoS(wsID, QoS, value): updates the defined QoS attribute value of a given service.
5. addService(wsdlURL, owlURL, wslaURL): adds a new service into the internal data structures.
6. removeService(wsID): removes a service from the internal data structures.

The *Bootstrap subsystem* is coordinated by the *System initializer*. Via the initialize interface, this component is directed to start the bootstrap process realizing the preprocessing. During it, the *System initializer* step by step calls the *WS Reader*, *Data structure builder*, and *Compatibility evaluator*. *WS Reader* is responsible for reading the service description documents (WSDLs, OWLSs, and WSLAs). It parses these documents and the read information are processed by the *WS processor*. After the basic analysis of web services is done (including e.g. the identification of the user data dependent services), the basic data structures are built. Then, the *Compatibility evaluator* evaluates which web services are compatible. After the bootstrap finishes, the system is ready to compose the web services which are actually available.

The *User querying subsystem* is managed by the *Process manager*. Via the *startQuery* interface, this central component provides the composition functionality. It is also responsible for calling a call back web service submitting the resulted composition. After receiving the composition query, the *Process manager* puts it into a corresponding queue. Based on the behavior depicted in Fig. 2, immediately when the composition could start, the *Composition realizer* runs the composition. It runs parallel threads to execute the selection of unusable and usable services. Each process operates over data structures managed by the *Data structure manager*. After the solution is found,

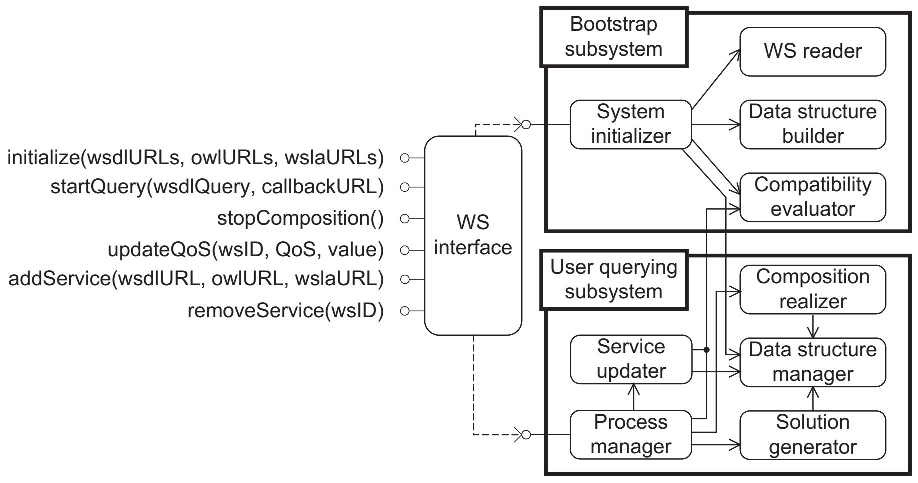

Figure 5: Composition system architecture.

the *Solution generator* retrieves the solution from the data structure and serializes it into the required format (e.g. BPEL). After this, the call back web service is invoked to submit the solution.

If an update request is received via some of the dedicated interfaces, the *Process manager* puts it into a corresponding queue. Immediately as it is possible, the *Service updater* processes the request. It involves the *Compatibility evaluator* (only in the case of adding a service) and *Data structure manager* to realize the required changes in the data structures.

The composition system can be used in two manners. First, its functionality can be accessed via a Web service providing operations corresponding to the defined interfaces. In the second case, the composition system implements a JAVA application programming interface (API) usable within another software system. In this case, the interfaces correspond to class methods (in the sense of object oriented paradigm).

5. Contributions

The contributions of our work, according to the defined objectives of the thesis, are summarized in the following points:

1. precise definition of pre-/post-conditions of Web services and condition compatibility evaluation [5, 7, 4],
2. feasible pre-/post-condition aware service composition [5, 7, 4],
3. fast and scalable QoS-driven control-/data-flow design of the composite service [6, 7, 8],
4. prompt updates of the internal data structures due changes in the service environment [8],
5. software design of the composition system [6],
6. examination of the overhead in the composition time due continuing composition query arrival [8],
7. evaluation of the effect of frequent changes in the service environment [8],

Regarding the *Functional aspects of Web services*, we contribute by point 1) and 2). We show that the pre-/post-condition aware service composition, as formally defined in the thesis, is feasible.

The *QoS optimization* is covered in point 3). Our approach proves that the requirement to optimize the exploratory Web service composition, according to the QoS, could be satisfied.

Effective handling of the *Changes in the service environment* is satisfied by point 4) and 7). We confirm that it is possible to effectively manage the changes of the service environment, requiring a change in the internal data structures used during the composition process. Moreover, this is possible to achieve also when considering continuing arrival of update and composition queries, which can be thus processed without unacceptable delay.

The *Effectiveness* is a general objective of our work and each of our contributions concerns this issue. Our results show that the performance issue can be handled in the context of each additional (functional) requirement. It is possible to realize automatic dynamic Web service composition in acceptable time, even if the QoS and pre-/post-condition are considered, the queries arrive continuously, and we must also react to changes in the service environment. Considering the effectiveness of condition compatibility evaluation, we proposed two approaches, based on a relational database and encoding, which show to be good performing even for complex formulae. In the context of QoS-driven control-/data-flow design, we developed a method of service space restriction, which could be applied also in other approaches, and showed significant improvement in performance when it is applied.

The objective to develop a *Composition system* relates to the point 5). To satisfy all the (non)functional requirements to Web service composition, we designed our composition tool as a queuing system. It consists from several modules, realizing individual tasks, which are coordinated to avoid inconsistency and incorrect results.

How we deal with the *Continuing user query arrival* is addressed by contribution 6). We showed that the natural overhead, caused by multiple tasks requiring processing, is in our approach low enough and the processing time remains acceptable.

All the work presented in this thesis is done with strong focus on the effectiveness. The aim was to propose solutions which can be applied even in large scale problems. One of the most influential issue, but not the only one, is the number of Web services which are considered during the composition. To make an imagination about the number of currently available Web services, we made an overview of some public repositories. The number of services which can be found in these repositories is from some thousands up to some ten thousands. The largest one, called *Seekda*[7], included at August 2010 more than 28 000 service descriptions, as claimed by the repository provider. To process this number of services, the performance of the approaches must be really good. Moreover, since the number of available services rises, scalability is an important isssue too.

Our work contributes by an approach to effective QoS driven composition showing promising results in terms of effectiveness and scalability [6, 7, 8]. We have been experimenting with different data sets, created by a third party tool, consisting from 10 000 up to 100 000 services. Our composition system was able to process these service sets and presented good performance in any of the test cases. The performance scales well depending on the size of the service repository. Even in the hardest case, the composition time is less than 200 msecs. From the user point of view, we consider this time as acceptable.

The fact that our approach performs well is supported also by achievement of good results at the *Web Services Challenge 2009*. *Web Services Challenge* is a world competition aimed at developing software components and/or intelligent agents that have the ability to discover pertinent Web services and also compose them to create higher-level functionality[8]. The competition presents an international forum where research groups from all over the world can test and compare their results. It provides a unique opportunity to evaluate different approaches under common conditions. To participate the competition, an implementation fulfilling the defined requirements must be provided for the organizers, so they can run and evaluate it. In 2009 the Web Service Challenge focused on automatic Web service composition considering the QoS [22]. The approaches were evaluated at two levels. First, the performance was tested. Second, the composition systems were evaluated from the architecture point of view by a committee.

Considering performance, even during the hardest test set at the competition, consisting from 15 000 services, our implementation of a composition system found a solution in acceptable time – in less than 300 msecs. The time includes a call of the composition system from a client application, composition, transformation of the result into BPEL format, and realization of a callback from the composition system to submit the result back to the client application. We achieved the third place in this category. The two approaches, scored better than our, performed better considering the first four test sets. In the hardest test set, one of the approaches showed an increase in the composition time to more than 900 msecs. From this it can be assumed that the approach does not scale well regarding the number of services. The rest of the approaches presented problems even with the easier test sets. Most of them did not found the solution in a given time limit (5 minutes) even for the basic test cases.

Our approach which took part at the challenge was only a preliminary version of the current one. We were able to improve its performance in more that one order of magnitude considering the composition time. Due this, we believe that the our approach is one of the best performing from any known ones.

In the architecture category, our approach won the competition. Our composition system was considered as the best, having additional features making it more practically applicable.

To make our approach more closer to practical usage, we dealt also with handling changes in the Web service environment. We considered the change of the QoS attribute values, adding a new service, and removing a service from a service repository. These changes are a natural phenomena and its crucial to be able to manage them effectively. Only a little research attention was devoted to these issues. We have designed an algorithm performing updates in our data structures based on the changes in the service environment. Our experiments show that the changes are handled quickly. This is important to assure that the composition reflects the current situation in the service set and the composition time is not significantly affected.

The performance of our composition system was evaluated also in a scenario when multiple queries may be independently sent to the composition system. These are collected in a queue and processed one by one. Composition and update queries are considered, each collected in a separate queue. Hence, our composition system can be modelled as a queuing system with different type of requests. We have analyzed how the system behaves when queries are sent with different frequencies. The results show that the system can manage and remain stable also if the queries arrive frequently, e.g. 100 per a second.

6. Conclusions

The aim of our thesis was to contribute to the field of automatic dynamic semantic Web service composition. It focuses on service description able to capture its functionality more precisely. This is done by defining the pre-/post-conditions of services in terms of predicate logic formulae. The approach achieves much better results in the way of more meaningful compositions. Without consideration of the conditions, the composition may result in an inappropriate solution, not satisfying the user goal from functionality point of view. Beside the functional aspect of the service composition, we dealt also with the non-functional properties. These are important too, to achieve user satisfaction. In this thesis we had shown, that it is possible to realize a pre-/post-condition aware, QoS driven composition, even if large-scale service repositories are considered. This is a crucial issue to make the overall service composition idea a reality. Moreover, in practical scenarios, the service composition must deal with the changes in the service environment and be able the process multiple queries from different users. We had shown that these issues can be handled effectively too. Our work was presented and published at reputable conferences [3, 4, 5, 6, 7, 8, 18].

[7]`http://Webservices.seekda.com/`
[8]`http://ws-challenge.georgetown.edu/wsc09`

Our approach can be used to solve the part of the whole Web service composition problem. To achieve practical applicability, further issues must be addressed and solved. The current research shows that the basic idea of Web service composition is vigorous. However, it focuses only to a part of the whole problem. Real use cases must be identified and studied to realize what exactly are the further requirements, which are necessary to deal with in practical scenarios.

On the other side, there are still problems requiring attention. We believe that the developed methods are already advanced enough to bring benefits when applied in real life. The essential thing is to provide tools, frameworks, and guidelines supporting the whole life-cycle of Web services and composition systems. These must make convenient different processes such as the software design of the Web services [30], their semantic annotation, and provision of the composition capabilities [37, 13].

Acknowledgements. This work was partially supported by the Slovak Research and Development Agency under the contract No. APVV-0391-06, the Scientific Grant Agency of SR, grant No. VG1/0508/09 and it is a partial result of the Research & Development Operational Program for the project Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 26240120029, co-funded by ERDF. We would like to thank to the organizers of the WS-Challenge 2009 to provide a great forum progressing the research of Web services composition.

References

[1] I. Adan and J. Resing. *Queueing Theory*. Eindhoven University of Technology, 180 pages, 2002. `http://www.win.tue.nl/\%7Eiadan/queueing.pdf`, downloaded in April, 2010.

[2] M. Alrifai, T. Risse, P. Dolog, and W. Nejdl. A scalable approach for qos-based web service selection. In *Service-Oriented Computing 2008 Workshops*, pages 190–199. Springer-Verlag, 2009.

[3] P. Bartalos and M. Bieliková. Enhancing semantic web services composition with user interaction. In *SCC '08: Proc. of the 2008 IEEE Int. Conf. on Services Computing*, pages 503–506. IEEE CS, 2008.

[4] P. Bartalos and M. Bielikova. Adapting i/o parameters of web services to enhance composition. In *NWESP '09. Fifth International Conference on Next Generation Web Services Practices, 2009.*, pages 17 –22, 2009.

[5] P. Bartalos and M. Bieliková. Fast and scalable semantic web service composition approach considering complex pre/postconditions. In *WSCA '09: Proc. of the 2009 IEEE Congress on Services, Int. Workshop on Web Service Composition and Adaptation*, pages 414–421. IEEE CS, 2009.

[6] P. Bartalos and M. Bieliková. Semantic web service composition framework based on parallel processing. In *Int. Conf. on E-Commerce Technology 2009*, pages 495–498. IEEE CS, 2009.

[7] P. Bartalos and M. Bieliková. Qos aware semantic web service composition approach considering pre/postconditions. In *Int. Conf. on Web Services 2010*, pages 345–352. IEEE CS, 2010.

[8] P. Bartalos and M. Bieliková. Effective qos aware web service composition in dynamic environment. In *Int. Conf. on Information Systems Development 2010*. Springer, 2010, Accepted.

[9] U. Bellur and R. Kulkarni. Improved matchmaking algorithm for semantic web services based on bipartite graph matching. *IEEE International Conference on Web Services*, 0:86–93, 2007.

[10] U. Bellur and H. Vadodaria. On extending semantic matchmaking to include preconditions and effects. In *ICWS '08: Proceedings of the 2008 IEEE International Conference on Web Services*, pages 120–128, Washington, DC, USA, 2008. IEEE Computer Society.

[11] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. Heuristics for qos-aware web service composition. In *ICWS '06: IEEE Int. Conf. on Web Services*, pages 72–82. IEEE CS, 2006.

[12] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 34–43, 2001.

[13] M. B. Blake, W. Tan, and F. Rosenberg. Composition as a service. *IEEE Internet Computing*, 14(1):78–82, 2010.

[14] J. Cardoso and A. P. Sheth. *Semantic Web Services, Processes and Applications*. Springer, 2006.

[15] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana. The next step in web services. *Commun. ACM*, 46(10):29–34, 2003.

[16] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer Verlag, Berlin Heidelberg, 2007.

[17] K. Fujii and T. Suda. Semantics-based dynamic service composition. *IEEE Journal on Selected Areas in Communications*, 23(12):2361 – 2372, dec. 2005.

[18] O. Habala, M. Paralič, V. Rozinajová, and P. Bartalos. Semantically-aided data-aware service workflow composition. In *SOFSEM '09: Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, pages 317–328, Berlin, Heidelberg, 2009. Springer-Verlag.

[19] Z. Huang, W. Jiang, S. Hu, and Z. Liu. Effective pruning algorithm for qos-aware service composition. In *Int. Conf. on E-Commerce Technology 2009*, pages 519–522. IEEE CS, 2009.

[20] W. Jiang, C. Zhang, Z. Huang, M. Chen, S. Hu, and Z. Liu. Qsynth: A tool for qos-aware automatic service composition. In *Int. Conf. on Web Services 2010*, pages 42–49. IEEE CS, 2010.

[21] M. Klusch, A. Gerber, and M. Schmidt. Semantic web service composition planning with owls-xplan. In *AAAI Fall Symposium on Semantic Web and Agents*, pages 55–62, Arlington VA, USA, 2005. AAAI Press.

[22] S. Kona, A. Bansal, B. Blake, S. Bleul, and T. Weise. A quality of service-oriented web services challenge. In *Int. Conf. on E-Commerce Technology 2009*, pages 487–490. IEEE CS, 2009.

[23] S. Kona, A. Bansal, M. B. Blake, and G. Gupta. Generalized semantics-based service composition. In *ICWS '08: Proc. of the 2008 IEEE Int. Conf. on Web Services*, pages 219–227. IEEE CS, 2008.

[24] S. Kona, A. Bansal, and G. Gupta. Automatic composition of semantic web services. In *ICWS '07: Proc. of the 2007 IEEE Int. Conf. on Web Services*, pages 150–158. IEEE CS, 2007.

[25] S. Kona, A. Bansal, L. Simon, A. Mallya, G. Gupta, and T. D. Hite. Usdl: A service-semantics description language for automatic service discovery and composition. *Int. J. Web Service Research*, 6(1):20–48, 2009.

[26] F. Lécué and N. Mehandjiev. Towards scalability of quality driven semantic web service composition. In *Int. Conf. on Web Services 2009*, pages 469–476. IEEE CS, 2009.

[27] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[28] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. Shop2: An htn planning system. *J. Artif. Intell. Res. (JAIR)*, 20:379–404, 2003.

[29] OWL services coalition. *OWL-S: semantic markup for web services*. OWL-S coalition, 2004. retrieved from http://www.daml.org/services/owl-s/1.0/owl-s.html, 2007.

[30] M. P. Papazoglou and J. Yang. Design methodology for web services and business processes. In *TES '02: Proceedings of the Third International Workshop on Technologies for E-Services*, pages 54–64, London, UK, 2002. Springer-Verlag.

[31] T. Payne and O. Lassila. Guest editors' introduction: Semantic web services. *IEEE Intelligent Systems*, 19(4):14–15, 2004.

[32] J. Peer. *Web Service Composition as AI Planning – a Survey*. University of St.Gallen, 2005.

[33] P. Rajasekaran, J. Miller, K. Verma, and A. Sheth. Enhancing web services description and discovery to facilitate composition. In *Proceedings of the 1st Int. Workshop on Semantic Web services and Web Process Composition*, pages 34–47. LNCS, 2004.

[34] J. Rao and X. Su. A survey of automated web service composition methods. *Semantic Web Services and Web Process Composition*, 3387/2005:43–54, 2005.

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

[36] F. Rosenberg, P. Celikovic, A. Michlmayr, P. Leitner, and S. Dustdar. An end-to-end approach for qos-aware service composition. *IEEE International Enterprise Distributed Object Computing Conference*, pages 151–160, 2009.

[37] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar. Towards composition as a service - a quality of service driven approach. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 1733–1740, Washington, DC, USA, 2009. IEEE Computer Society.

[38] F. Rosenberg, M. B. Muller, P. Leitner, A. Michlmayr, A. Bouguettaya, and S. Dustdar. Metaheuristic optimization of large-scale qos-aware service compositions. *IEEE International Conference on Services Computing*, pages 97–104, 2010.

[39] E. Sirin, J. Hendler, and B. Parsia. Semi-automatic composition of web services using semantic descriptions. In *Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003*, 2002.

[40] B. Srivastava and J. Koehler. Web service composition – current solutions and open problems. In *Int. Conf. on Automated Planning and Scheduling 2003*, 2003.

[41] R. Studer, S. Grimm, and A. Abecker. *Semantic Web Services: Concepts, Technologies, and Applications*. Springer, 2007.

[42] A. Urbieta, E. Azketa, I. Gomez, J. Parra, and N. Arana. Analysis of effects- and preconditions-based service representation in ubiquitous computing environments. In *International Conference on Semantic Computing*, pages 378–385, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[43] G. Wang, D. Xu, Y. Qi, and D. Hou. A semantic match algorithm for web services based on improved semantic distance. In *NWESP '08: Proceedings of the 2008 4th International Conference on Next Generation Web Services Practices*, pages 101–106, Washington, DC, USA, 2008. IEEE Computer Society.

[44] S. E. Wang, Y. Semantic structure matching for assessing web-service similarity. In *ICSOC '03: Proceedings of First International Conference on Service Oriented Computing*, pages 194—-207, Berlin, 2003. Springer.

[45] A. B. Williams, A. Padmanabhan, and M. B. Blake. Experimentation with local consensus ontologies with implications for automated service composition. *IEEE Trans. on Knowledge and Data Engineering*, 17:969–981, 2005.

[46] P. K. Yoon, C.-L. Hwang, and K. Yoon. *Multiple Attribute Decision Making: An Introduction (Quantitative Applications in the Social Sciences)*. Sage Publications Inc, 1995.

[47] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *Int. Conf. on World Wide Web 2003*, pages 411–421. ACM, 2003.

[48] L. Zeng, B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.

Selected Papers by the Author

Bartalos, P., Bieliková, M.: QoS Aware Semantic Web Service Composition Approach Considering Pre/Postconditions. In: *ICWS '10: Int. Conf. on Web Services 2010*, IEEE Computer Society Press, (2010), pp. 345–352.

Bartalos, P., Bieliková, M.: Effective QoS aware web service composition in dynamic environment. In: *ISD '10: Int. Conf. on Information Systems Development 2010*, Springer, (2010), To appear.

Bartalos, P., Bieliková, M.: Composition and undesired web service execution effects. In: *WoSS 2010: Proc. of 12th Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Workshops on Software Services, Timisoara, Romania*, (2010).

Bartalos, P., Bieliková, M.: Semantic Web Service Composition Framework Based on Parallel Processing. In: *CEC '09: Int. Conf. on E-Commerce Technology 2009*, IEEE Computer Society Press, (2009), pp. 495–498.

Bartalos, P., Bieliková, M.: Fast and Scalable Semantic Web Service Composition Approach Considering Complex Pre/Postconditions. In: *Services/WSCA '09: IEEE Congress on Services, Int. Workshop on Web Service Composition and Adaptation 2009*, IEEE Computer Society Press, (2009), pp. 414–421.

Bartalos, P., Bieliková, M.: Adapting I/O Parameters of Web Services to Enhance Composition. In: *NWeSP '09: Fifth Int. Conf. on Next Generation web Services Practices 2009*, IEEE Computer Society Press, (2009), pp. 17–22.

Habala, O., Paralič, M., Rozinajová, V., Bartalos, P.: Semantically-Aided Data-Aware Service Workflow Composition. In *SOFSEM '09: Theory and Practice of Computer Science 2009*, Springer, LNCS, (2009), pp. 317–328.

Bartalos, P., Bieliková, M.: Enhancing semantic web services composition with user interaction. In *SCC '08: IEEE Int. Conf. on Services Computing 2008*, IEEE Computer Society Press, (2008), pp. 503–506.

Bartalos, P., Bieliková, M.: An approach to object-ontology mapping. In: *CEE-SET '07: 2nd IFIP Central and East European Conf. on Software Engineering Techniques 2007*, (2007), pp. 67–79.

Bartalos, P., Barla, M., Frivolt, G., Tvarožek, M., Andrejko, A., Bieliková, M., Návrat, P.: Building an Ontological Base for Experimental Evaluation of Semantic Web Applications. In *SOFSEM '07: Conference on Current Trends in Theory and Practice of Computer Science 2007*, Springer, LNCS, (2007), pp. 682–692.

Barla, M., Bartalos, P., Bieliková, M., Filkorn, R., Tvarožek, M.: Adaptive portal Framework for Semantic Web applications. In: *AEWSE '07: 2nd Int. Workshop on Adaptation and Evolution in Web Systems Engineering 2007*, (2007), pp. 87–93.

Barla, M., Bartalos, P., Sivák, P., Szobi, K., Tvarožek, M., Filkorn, R.: Ontology as an Information Base for Domain Oriented Portal Solutions. In: *ISD '06: 15th Int. Conf. on Information Systems Development 2006*, Springer, (2006), pp. 423–433.

Bartalos, P.: Semantic Web Services. In: Studies on selected topics of software and information systems 4 : Advanced methods of software design. Advanced methods for acquisition, search, representation, and presentation of information. STU Bratislava, 2009, ISBN 978-80-227-3139-3, pp. 167–199. (In slovak)

Bartalos, P., Paralič, M., Habala, O., Hluchý, L., Rozinajová, V., Gažák, M.: SEMCO-WS: Semantic Composition of Web Services. In: *Znalosti 2010, Jindřichův Hradec*, VŠE Prague, (2010), pp. 293–296.

Paralic, M., Habala, O., Paralic, J., Bartalos, P.: Semantic Composition of Web and Grid Services. In: *Znalosti 2009, Brno*, STU Bratislava, (2009), pp. 355–358.

Bartalos, P., Gažák, M.: Semantic web services based crisis information system exploiting automated workflow composition. In: *GCCP '09: 5th Int. Workshop on Grid Computing for Complex Problems 2009*, VEDA, (2009), pp. 44–51.

Kapustík, I., Rozinajová, V., Bartalos, P.: Supporting user collaboration by enhancing semantics based communication tool. In: *WIKT '08: 3rd Workshop on Intelligent and Knowledge Oriented Technologies 2008*, STU Bratislava, (2008), pp. 13–16. (In slovak)

Bartalos, P., Kapustik, I., Rozinajova, V.: Visual support of workflow composition involving collaboration. In: *GCCP '08: Int. Workshop on Grid Computing for Complex Problems 2008*, SAS, (2008), pp. 120–127.

Bartalos, P., Bieliková, M.: Data-aware Composition of Workflows of Web and Grid Services. In: *Cracow Grid Workshop ´08*, (2009), pp. 120–128.

Bartalos, P., Bieliková, M.: (S)CRUD Pattern Support for Semantic Web Applications. In: *SOFSEM '08: Theory and Practice of Computer Science. Volume II - Student Research Forum : 34th Conference on Current Trends in Theory and Practice of Computer Science*, Šafárik University Košice, (2008), pp. 10–21.

Bartalos, P., Bieliková, M.: Semantic web services supported by collaboration. In: *WIKT '07: 2nd Workshop on Intelligent and Knowledge oriented Technologies 2007*, TU Košice, (2007), pp. 67–70. (In slovak)