eng. Viorica Rozina CHIFU

PhD. Thesis

Automatic Web Service Composition

Abstract

Advisor: Prof. Dr. eng. Ioan SALOMIE

Cluj-Napoca, 2010
List of Papers

Papers indexed DBLP


**Papers indexed IEEE Xplore**


Papers indexed ACM


Journal Articles


Conference Papers


Content

Introduction.................................................................................................................................................. 1
1.1 Semantic Web Services.................................................................................................................................. 1
1.2 Web Service Composition................................................................................................................................ 2
  1.2.1 Complexity of the automatic Web service composition............................................................................ 4
1.3 Thesis objectives........................................................................................................................................... 5
1.4. Thesis structure........................................................................................................................................ 6
Related work.................................................................................................................................................. 10
  2.1 S Semantic Web Services............................................................................................................................ 10
    2.1.1 SAWSDL............................................................................................................................................... 10
    2.1.2 WSDL–S............................................................................................................................................... 11
    2.1.3 OWL–S............................................................................................................................................... 12
    2.1.4 WSMO............................................................................................................................................... 12
  2.2 State of the art in the field of Web service composition.................................................................................. 13
    2.2.1 Web service composition based on the workflow technique........................................................................ 14
    2.2.2 Web service composition using AI planning............................................................................................ 15
    2.2.3 Web service composition based on formal methods................................................................................... 21
    2.2.4 Ontology driven Web service composition............................................................................................... 23
    2.2.5 Graph-based Web Service composition................................................................................................. 24
    2.2.6 Biologically inspired Web service composition........................................................................................ 25
Web service composition technique based on fluent calculus............................................................................ 28
  3.1 Fluent Calculus and FLUX.......................................................................................................................... 28
    3.1.2 Fluent Calculus versus Situation Calculus................................................................................................. 32
  3.2 Translating from OWL-S ontology............................................................................................................. 33
    3.2.1 OWL-S Ontology................................................................................................................................... 33
    3.2.2 Translating the OWL-S atomic process into the fluent calculus................................................................. 34
    3.2.3 Translating the OWL-S compose process into the fluent calculus............................................................... 35
  3.3 Translating from the domain ontology into the fluent calculus........................................................................ 48
    3.3.1 A Web service domain ontology................................................................................................................ 49
    3.3.2 Translating a domain ontology into the fluent calculs................................................................................ 53
3.4 Web service composition using the fluent calculus

3.4.1 The composition algorithm

3.5. The experimental prototype

3.5.1 The architecture of the experimental prototype

3.5.2 Experimental Results

3.6 Comparison with other similar works

3.7 Conclusion

A method for automatic Web service discovery

4.1 The semantic similarity

4.1.1 The degree of semantic match (DoM)

4.2 Evaluating the degree of match between two services

4.2.1 Encoding the semantic Web service descriptions

4.2.2 Metrics for evaluating the degree of match between two services

4.3 ACDOM – The algorithm for computing the degree of match

4.3.1 Metrics for evaluating the performances of the ACDOM algorithm

4.4 An example of computing the DoM between two services

4.5 The experimental prototype

4.5.1 The architecture of the experimental prototype

4.5.2 Experimental results

4.6 Conclusions

A Web service composition technique based on a graph of service cells

5.1 Problem formalization

5.2 The graph building algorithm

5.3 The composition algorithm

5.3.1 The initialization phase

5.3.2 The search phase

5.3.3 The partial execution phase

5.3.4 The selection and execution phase

5.4 SGDL – The service graph description language

5.5 The experimental prototype

Immune-inspired Web service composition technique

6.1 The enhanced planning graph for the Web service composition

6.1.1 Definitions

6.1.2 The enhanced planning graph model for the Web service composition
1. Introduction

Web services promise to turn the Web from a static collection of documents into a vast library of programs. This is the main reason why the notion of service is of a considerable interest from both the industry and the academic research. The Web Service technology has opened the door towards a new era dominated by applications with a high degree of intelligence, capable of making decisions and searching for information on the Internet [4]. The current standard technologies for Web services (WSDL, UDDI) provide descriptions only at the syntactic level of their functionality, without any formal description of their semantics. The lack of any machine interpretable semantics requires human intervention for service discovery and composition. This drawback prevents the use of Web services in complex business contexts, where the automation of these business processes is necessary. Semantic Web Services enhance the existing standards by annotating the services with semantic descriptions provided by ontologies. The semantic descriptions are necessary in order to allow for automatic Web service composition. In an automatic composition system, the users’s role is limited to specifying the functional requirements. Instead, the system should define the data and control flow by assembling individual services based on user provided inputs and expected outputs.

The research field of automatic service composition is a top research area, of great actuality in information technology worldwide and nationwide. The automatic Web service composition is crucial for the success of the enterprise application integration (eai) and of the integration of business-to-business (b2b) applications on the internet.

The objective of this thesis was the study and elaboration of a set of techniques for ontology driven automatic Web service composition. For this objective, we will study, elaborate, and experimentally validate the models, methods, and algorithms of automatic Web service composition, aiming at: automatic service discovery, the development of algorithms for service selection based on quality of service (QoS) criteria, automatic service composition and automatic service selection based on quality of service (QoS) criteria. In order to test, assess, and validate the proposed composition techniques, we implement the experimental prototypes for automatic Web service discovery and composition which will also allow a comparison with other existing approaches of the same kind.

The main ideas from this thesis have been published in 32 articles, nine of which have been indexed DBLP.

2. State of the art in automatic Web service composition

This chapter presents the state of the art in the automatic Web service composition. The chapter starts by presenting the recent efforts to bring semantics to Web services, including initiatives, projects, and languages (i.e. OWL-S, WSMO, WSDL-S) followed by a review of some existing researches in the domain of automatic Web service composition. Based on the domain analysis we have identified six types of techniques for automatic Web service composition:

- Web service composition using the workflow technique;
- Web service composition using AI Planning;
  - Web service composition using situation calculus
  - Web service composition using rule-based planning
  - Web service composition using other AI planning techniques
  - Web service composition using theorem proving
- Web service composition based on formal methods;
- Ontology driven Web service composition;
- Graph-based Web service composition;
- Biologically inspired Web service composition
  - Web Service Composition based on ant principles;
  - Web Service Composition based on particle swarm optimization;
  - Web Service Composition based on immune principles;

For each of these composition techniques, we present same approaches existing in the domain, as well as the advantages and disadvantages of these approaches.

3. A Web service composition technique based on fluent calculus

This chapter presents a new technique for automatic Web service composition based on the fluent calculus formalism [7, 8, 9, 10, 11]. The proposed technique consists of a translation method of the semantic Web service description into the fluent calculus as well as a Web service composition method based on the fluent calculus. In our approach, the semantic descriptions of Web services (i.e. OWL-S [13] or SAWSDL [14]) are translated into the state update axioms and action precondition axioms in the fluent calculus. The state update axioms specify which fluents are changed by an action and under which circumstances. The action precondition axioms specify the necessary conditions under which an action is possible to be executed. The knowledge base resulted from the translation process is used by the service composition method to identify a Web service composition which satisfies a complex user request. The presented composition method has the roots in the AI planning domain and takes the advantage of the fluent calculus formalism that uses progression as the reasoning mode. To test our approach, we have implemented an experimental prototype which covers the following stages of the composition process: captures the user request, generates a solution for the semantic Web service composition by using the fluent calculus and finally, tests and validates the solution by means of the LTSA execution engine. The composition flow is automatically derived, without the need for any manual intervention. As main compositional control structures our prototype uses sequential and
concurrent operations. Our experimental prototype has been tested on a set of scenarios from the trip planning and the social events planning domains.

The original contributions of this chapter are: (i) a method for translating the semantic Web service description (OWL-S, SAWSDL) into the fluent calculus formalism; (ii) a method for automatic Web service composition using the fluent calculus; and (iii) an experimental prototype for validating/testing the proposed approach.

4. A method for automatic Web service discovery

This chapter presents a new method for semantic Web service discovery. To evaluate the semantic matching between two service descriptions we have developed a service matching algorithm. Our matching algorithm is based on the symmetric learning accuracy [HS98], an evaluation metric from the domain of ontology learning and population. The symmetric learning accuracy measure is adapted/used to express the degree of semantic match between a service request and a service advertisement. The matching algorithm supports flexible matching (i.e. exact, plug-in, and subsumes) between the advertised services and the service request. In order to allow for an efficient evaluation of the semantic matching between a service request and a service advertisement, we employ an encoding of the service descriptions based on labeling schemes [1]. The evaluation of the semantic matching between a service request and an advertised service is based on concepts, their semantic relations, their common and distinguishing properties, and the semantic relations between their properties. The semantic matching is computed by considering the semantic information (of the Web service request and Web service advertisements) encoded in the common domain ontology.

The discovery method was tested and validated on a set of semantic Web services included in the OWL-STCv3 [12]. The OWL-STCv3 is an OWL-S service retrieval test collection, proposed for evaluating the performances of service matchmaking algorithms. This OWL-STCv3 service collection contains:

- a set of 1007 Web services semantically described in OWL-S;
- a set of ontologies from seven different domains (ex.: education, medical care, food, travel, communication, economy and weapon);
- a set of 28 queries.

The accuracy of the proposed discovery method is computed by using the information retrieval measures of recall, precision. Precision for a query was defined as the number of relevant services retrieved for a query divided by the total number of services retrieved for that query, and recall was defined as the number of relevant services retrieved for a query divided by the total number of existing services (which should have been retrieved).

Figure 4.1 depicts the recall/precision values computed based on the experimental results obtained by testing our method on all the 28 queries from OWLSTCv3. The mean values of the recall and precision computed for all the queries are: $\text{precision}_{\text{mean}} = 0.801785714$, $\text{recall}_{\text{mean}} = 0.959286$. 
The main advantages of the proposed discovery method are presented in what follows. The service matchmaking is based on evaluating the semantic similarity of ontology classes, by taking advantage of both the class hierarchy and the properties of the classes. One major benefit over keyword based matching is the use of explicit semantics. These explicit semantics are available through the domain ontology of the semantic Web services. The advantage of our approach is related to the type of the degree of match. Instead of attaching a discrete value for every semantic Web service, the method labels each of the advertisements with a continuous value from the interval $[0, 1]$. This property is useful when dealing with a large number of advertised Web services. Also our approach evaluates efficiently the degree of match between a service request and a service advertisement. The efficiency is achieved by employing a suitable encoding of the semantic service descriptions based on labeling schemes [1].

The original contributions of this chapter are: (i) a set of semantic similarity metrics for evaluating the degree of match between two services; (ii) a semantic matching algorithm based on the proposed metrics; (iii) an experimental prototype for validating/testing the proposed approach. Also, the obtained experimental results demonstrate the validity of the proposed approach.

5. Web service composition technique based on a graph of service cells

This chapter proposes a new method for the automatic composition of semantic Web services. The proposed method consists in constructing a directed acyclic graph of service
cell and then searching for sub-graphs of this graph, which will represent the candidate composition solutions.

First, the method constructs a directed acyclic graph of service cell. The construction of the graph takes into account the semantic matching between the output parameters of some services and the input parameters of other services. For checking the semantic matching between an output concept of a service $S_i$ and an input concept of a service $S_j$, three types of semantic matching are considered: (i) exact – the output parameter of the service $S_i$ and the input parameter of the service $S_j$ are equivalent; (ii) plug-in – the output parameter of the service $S_i$ is subsumed by the input parameter of the service $S_j$; (iii) subsume – the output parameter of the service $S_i$ subsumes the input parameter of the service $S_j$. This semantic matching is computed based on the subsumption relation between concepts according to a common ontology used for the semantic description of services.

Second, composition sub-graphs that satisfy the functionality requested by a client are searched for and ranked according to certain criteria such as user preferences and QoS. Starting from a given goal node of the graph, the searching for the sub-graphs is a reasoning and searching process based on backward chaining, similar to the behavior of a Prolog reasoning engine. The different sub-graphs found represent different service compositions.

Finally, the method selects the best service composition for a given user request. The best service composition is the directed acyclic sub-graph of optimal cost, given the set of user preference constraints.

To evaluate the method, an experimental framework that automatically composes services has been implemented. The effectiveness of the method has been demonstrated by using a set of scenarios from the trip planning domain.

We have proven by this composition method that the graph is a natural and well suited choice for representing Web service compositions. It is reasonable to state that each service can provide and receive data to or from a rather restricted set of other services, according to the semantics associated with its input and output ports. These matches are static in time and generally true for any composition, as long as the semantics remain the same. For this reason, it is useful to pre-compute these matches in a background process, invisible from the user's point of view. This approach brings a great speed improvement at execution level and transforms the composition problem into a rather straightforward path finding algorithm. The semantic knowledge and heuristics are concentrated into determining and ranking the compatibility connections in the graph. When determining the actual composition for a particular set of goals, the algorithm will search in the smallest domain theoretically possible, because it will consider only options which have already been established as meaningful and valid by the match making process.

In the proposed approach we have handled the problem of Web service composition from a practical and functional point of view, trying to identify the key requirements of such a system and find a consistent and direct way of solving them. We have also focused on the user and the way he/she would want to interact with the system. The goal was to create a feasible solution that could work in the real world. We tried to keep a level balance between the degree of automation in the process, and the degree of control and comfort a real user would feel while using the system. We think that such a system should work together with the user, acting as an adviser and recommending actions, but always letting the user to take the final decisions.
The original contributions of the chapter are: (i) the problem formalization; (ii) the method for automatic Web service composition based on a graph of service cells; (iii) a way to specify and integrate user preferences in the Web service composition process; (iv) a new formal language for storing the graph of the service cells, namely SGDL (service graph description language).

6. Immune-inspired Web service composition technique

This chapter presents an immune-inspired technique applied in the context of Web service composition to select the optimal composition solution. Our approach models Web service composition as a multi-layered process which creates a planning-graph structure along with a matrix of semantic links. We have enhanced the classical planning graph with the new concepts of service cluster and semantic similarity link. The semantic similarity links are defined between services on different graph layers and are stored in a matrix of semantic links. To calculate the degree of the semantic match between services, we have adapted the information retrieval measures of recall, precision and F_Measure [3]. The immune-inspired algorithm uses the enhanced planning graph and the matrix of semantic links to select the optimal composition solution employing the QoS attributes and the semantic quality as the selection criteria.

6.1 Problem formalization

Let be $\Gamma$ a domain ontology used for the semantic description of Web services and for building the user request.

**Definition 6.1 (service composition request):**

A service composition request is defined as a triple:

$$scr = (in, out, w)$$

where:

- $in \ {in_i | in_i \in \Gamma}$ – is the set of requested input parameters;
- $out = \ {out_i | out_i \in \Gamma}$ – is the set of requested output parameters;
- $w = \ (w_{QoS_{ind}}, w_{QoS_{glob}}, w_{sem}) \ | w_{QoS_{ind}}, w_{QoS_{glob}}, w_{sem} \in [0, 1]$ – is the set of user-provided weights.

**Note:**

1. $w_{QoS_{ind}} = \ {w_{QoS_1}, ..., w_{QoS_n}}$ - represents the user preferences regarding to the individual value of a QoS attribute (e.g.: price, running time, cost) for a composition solution
2. The $w_{QoS_{glob}}, w_{sem}$ weights represent the user preferences regarding to:
   2.1 The overall QoS value of a composition solution;
   2.2 The value of the semantic similarity score of a composition solution.
Definition 6.2 (Web service):
A Web service is defined as a triple:

\[ ws = (f, in, out) \]

where:
- \( f \in \Gamma \) – represents an ontology concept that semantically describe the service operation;
- \( s.in = \{ s.in_i \mid s.in_i \in \Gamma \} \) – represents the set of concepts that semantically describe the input parameters of the service operation;
- \( s.out = \{ s.out_i \mid s.out_i \in \Gamma \} \) – represents the set of concepts that semantically describe the output parameters of the service operation.

Definition 6.3 (virtual Web service):
We define a virtual service \( vs \) as a service that does not provide functionality and has only input or output parameters.

Remark 6.1: We consider two virtual services \( vs_{out} = (\emptyset, \emptyset, scr.in) \) and \( vs_{in} = (\emptyset, scr.out, \emptyset) \).

Note:
1. \( vs_{out} \) is a virtual service which has no inputs and functionality, only outputs. The outputs of the service \( vs_{out} \) represent the user provided inputs extracted from the service composition request \( scr \).
2. \( vs_{in} \) is a virtual service which has no outputs and functionality, only inputs. The inputs of the service \( vs_{in} \) represent the user expected outputs extracted from the service composition request \( scr \).

Definition 6.4 (service cluster):
We define a service cluster as:

\[ sc = \{ ws_k \mid \forall ws_i, ws_j \in sc, (ws_i, f \equiv ws_j, f \lor ws_i, f \subseteq ws_j, f \lor ws_i, f \supseteq ws_j, f) \land \\
(\forall ws_i, in_m \in ws_i, in, \forall ws_j, in_n \in ws_j, in \rightarrow ws_i, in_m \equiv ws_j, in_n \\
\lor ws_i, in_m \subseteq ws_j, in_n \lor ws_i, in_m \supseteq ws_j, in_n) \} \]

Note: A cluster of services groups concepts that semantically describe a set of services which have the same functionality and the same set of inputs parameters and are in one of the following relations: exact (\( \equiv \)), plug-in (\( c_i \subseteq c_j \)) or subsume (\( c_i \supseteq c_j \)).

Definition 6.5 (Literal):
We define a literal as:

\[ l = l_i \mid \forall vs_{in}, vs_{out}, ws_j \in WS, l_i \in \{ vs_i.in, vs_{o}.out, ws_j.in, ws_j.out \} \]

Note: A literal is a concept that semantically describes an input/output of a generic Web service \( ws_j \) or a virtual Web service \( vs_{in}, vs_{out} \).
**Definition 6.6 (Cluster of literals):**

We define a *cluster of literals* as:

\[ lc = \{ l_k | \forall l_i, l_j \in lc, \ l_i \equiv l_j \lor l_i \subseteq l_j \lor l_i \supseteq l_j \} \]

**Note:** A *cluster of literals* groups concepts that semantically describe the same input/output parameter of services. Between two concepts that semantically describe an input/output parameter of the services there are the following relations: exact \((c_i \equiv c_j)\), plug-in \((c_i \subseteq c_j)\) and subsume \((c_i \supseteq c_j)\).

**Definition 6.7 (Enhanced planning graph):**

We define an *enhanced planning graph* for the Web service composition as a set of tuple:

\[ EPG = \{(A_i, L_i)\} \]

where:

- \(i\) – represents the layer in the EPG;
- \(A_i = \{sc_j^i | sc_j^i \in i \}\) – represents the set of service clusters from the layer \(i\) in EPG;
- \(L_i = L_{i-1} \cup \{lc_m^i | lc_m^i \in i \}\) – represents the set of clusters of literals form the layer \(i\) in EPG.

**Note:** \((A_0, L_0)\) is a particular case where \(A_0 = \emptyset\) and \(L_0 = vs_{in}\), while \((A_n, L_n)\) is the last layer in the graph and \(vs_{out} \subseteq L_n\).

**Definition 6.8: (A composition solution):**

We define a composition solution satisfying a service request \(scr\) as:

\[ sol = \{ \forall i \in EPG \land \forall sc_j^i \in i \ (\exists! ws^i_{jk} \in sc_j^i | ws^i_{jk} \in sol) \} \]

where:

- \(i\) – represents the layer \(i\) in the EPG;
- \(sc_j^i\) – is the cluster \(j\) on the layer \(i\) in the EPG;
- \(ws^i_{jk}\) – is the service \(k\) in cluster \(j\) on the layer \(i\) in the EPG.

The construction of the *enhanced planning graph* is an iterative process, in which a new layer is added to the planning graph in each iteration. The services which contribute to the extension of the planning graph are provided by a discovery process that finds the appropriate Web services in a repository of services, based on the semantic matching between the services’ inputs and the set of literals corresponding to the previous layer.

The *Matrix of Semantic Links* (MSL) stores the semantic links established between the services on different layers. One Web service could be linked to several other Web services. We say that there is a semantic similarity link between two services \(s_1\) and \(s_2\) if there is a degree of match \((DoM)\) [5] between the set of output parameters of service \(s_1\) and the set of input parameters of service \(s_2\). The MSL is built iteratively during the composition process by using the information gathered during the service discovery process and the new information obtained during the *enhanced planning graph* construction. In MSL, both the
columns and the rows are labeled with services from the planning graph. A column and a row having the same index will be labeled with the same service.

**Definition 6.9:** (Semantic similarity link):

We define a semantic similarity link between two services \( ws_i, ws_j \) as \( sl_{ij} = \{ (ws_j.out_k, ws_p.in_l) \mid DoM(ws_j.out_k, ws_p.in_l) > 0, ws_j.out_k \in ws_j.out, ws_p.in_l \in ws_p.in \} \)

To compute the degree of the semantic match between services, we have adapted the well known information retrieval measures of recall, precision and \( F \_Measure \) [3]. We have considered three types of semantic matching between an output concept of a service \( s_i \) and an input concept of a service \( s_j \): exact, plug-in, subsume, and faild as presented in [5].

For finding the optimal composition we adapted and enhanced a version of the CLONALG algorithm [2], which was proposed for optimization problems. Our immune-inspired solution uses the enhanced composition planning graph \( EPG \), the matrix of semantic links \( MSL \), both generated as a result of the execution of the composition algorithm, and a multi-criteria function \( QF \) to find and rank the valid compositions according to \( QoS \) user preferences and semantic quality.

We tested and evaluated our composition and selection methods on a set of scenarios from the trip and social event attendance planning domains. The experimental results prove that a hybrid approach that combines an enhanced planning graph with an immune-inspired algorithm can be successfully applied for finding the optimal composition solution.

The original contributions of this chapter are: (i) the enhanced planning graph model, (ii) the metrics for evaluating the degree of semantic match between two services, (iii) the semantic similarity matrix, and (iv) the immune-inspired selection method for finding the optimal composition solution.

### 7. Final Conclusions

This chapter presents the thesis contributions and summarizes the future work from our own perspective. The thesis contributions are:

1. A state of the art in automatic Web service composition;
2. A technique for automatic Web service composition based on fluent calculus consisting of:
   - 2.1. A method for translating semantic Web service descriptions into the fluent calculus;
   - 2.2. A method for automatic Web service composition using fluent calculus;
   - 2.3. An experimental prototype for validating/testing the proposed approach;
3. A method for automatic Web service discovery consisting of:
   - 3.1. A set of metrics for evaluating the degree of semantic match between two services;
   - 3.2. An algorithm for automatic service discovery based on the proposed metrics;
   - 3.3. An experimental prototype for validating/testing the proposed approach;
4. A technique for automatic Web service composition based on a graph of service cells consisting of:
   4.1. Problem formalization;
   4.2. An algorithm for automatic service composition based on a graph of service cells;
   4.3. A new formal language for storing the graph of service cells;
   4.4. An experimental prototype for validating/testing the proposed approach;
5. An immune-inspired technique for automatic Web service composition consisting of:
   5.1. An enhanced planning graph model for the Web service composition;
   5.2. A matrix of semantic links;
   5.3. A method for automatic Web service composition based the enhanced planning graph model;
   5.4. An immune-inspired method for finding the optimal composition solution
   5.5 An experimental prototype for validating/testing the proposed approach

9. Selected Bibliography


