International Journal on Artificial Intelligence Tools
Vol. 24, No. 1 (2015) 1450015 (25 pages)

DOI: 10.1142/S0218213014500158

Semantically Aware Web Service Composition Through AI Planning

Ourania Hatzi* and Mara Nikolaidou†

Department of Informatics and Telematics, Harokopio University of Athens
9, Omirou str., 17778, Athens, Greece
**raniah@hua.gr*
†*mara@hua.gr*

Dimitris Vrakas‡ and Nick Bassiliades§

Department of Informatics, Aristotle University of Thessaloniki
54124, Thessaloniki, Greece
‡*dvrakas@csd.auth.gr*
§*nbassili@csd.auth.gr*

Dimosthenis Anagnostopoulos

Department of Informatics and Telematics, Harokopio University of Athens
9, Omirou str., 17778, Athens, Greece
dimosthe@hua.gr

Ioannis Vlahavas

Department of Informatics, Aristotle University of Thessaloniki
54124, Thessaloniki, Greece
vlahavas@csd.auth.gr

Received 4 June 2013
Accepted 19 June 2014
Published 23 February 2015

Web service composition is a significant problem as the number of available web services increases; however, manual composition is not an efficient option. Automated web service composition can be performed using AI Planning techniques, utilizing descriptions of available atomic web services, enhanced with semantic awareness and relaxation. This paper discusses a unified, semantically aware approach, handling both semantic (OWL-S & SAWSDL) and non-semantic (WSDL) web service descriptions. In the first case, ontology analysis is adopted to semantically enhance the planning domains and problems, in order to deal with cases where exact syntactic input-to-output matching is not feasible. In the non-semantic descriptions case, semantic information is acquired utilizing alternative sources such as lexical thesauri. Concept similarity measures are applied and utilized to achieve the desired degree of semantic relaxation. The solution to a web service composition problem is a plan describing the desired composite service. To support the proposed approach, the PORSCE framework has been implemented. The framework is modular, integrating

discrete web service description languages and semantic relaxation techniques. Based on the similarity measures suggested in the paper, performance issues are also explored.

Keywords: Semantic web services; intelligent web service composition; OWL-S; SAWSDL; WSDL; AI planning; PDDL; problem transformation; semantic relaxation.

1. Introduction

Web Services are designed to deal with the issue of interoperability between diverse software systems on the web,[2] by providing well-defined, standard interfaces[46] and means of communication.[42] Atomic web services offer specific, limited functionality, which in many cases does not meet user needs. More complex, enhanced functionality can be achieved through the combination of simple, atomic web services into composite ones. Manual composition, performed by selecting appropriate web services from a set of available ones, is hardly an efficient option. As the number of available web services continuously increases, locating and appropriately combining them involves significant complexity, resulting in impractical times.[23] The promising alternative is automated web service composition, where the composite service description is generated automatically, based on initial user requirements concerning both functional and non-functional properties.

Automated web service composition reaches its full potential in the Semantic Web, where the need for complex functionality is fulfilled by stating requirements at the semantic level, as opposed to syntactic descriptions. In order to accommodate semantics, a number of Semantic Web Service Languages have been introduced, such as SAWSDL (Semantic Annotations for WSDL)[39] and OWL-S,[30] leading to the notion of *Semantic Web Services*. SAWSDL in particular has been established as a W3C Recommendation, defining a mechanism to enable semantic annotations of web service descriptions in the WSDL standard.[46] Many tools and applications have been presented,[40,36] promoting the use of semantics in web service descriptions by facilitating creation, retrieval, management and exploitation of semantic annotations. The semantic web services paradigm is motivated by the fact that while the XML representation of the services characteristics in WSDL guarantees interoperability at the syntactic level, it is unable to capture the actual meaning of information, which would ensure semantic interoperability as well.[27] Enhancement of web service descriptions with semantics is essential for dynamic, automated web service discovery and composition.

The need for automated web service composition has triggered a number of research directions towards automated composition,[37] among which AI Planning proved to be very promising.[5] A prerequisite for enabling the use of planning algorithms is the transformation of the web service composition problem into a planning problem. Enhanced functionality, such as approximate composite services, can be provided by combining planning techniques with semantic information. Such information can be acquired either from web service semantic descriptions themselves (for example SAWSDL or OWL-S descriptions and the corresponding ontologies), or by alternative means, for the non-semantic descriptions case (WSDL descriptions), such as lexical thesauri.

Intervention to the composite service, by excluding or replacing certain atomic web services, as certain services might occasionally be unavailable or undesirable, is also important. Finally, as the use of multiple planners and semantic relaxation may produce a number of different composite web services satisfying the user requirements, accuracy assessment of each composite service is essential. However, a comprehensive framework for automated web service composition, satisfying key issues stated above, is still not available.

Research performed by the authors of this paper focuses on the field of composition of semantic web services. This paper attempts to take this research one step further, to semantic composition of web services, i.e. attempts to attach semantics to composition, even for web service descriptions with no inherent semantics. It introduces an integrated methodology for semantically aware, automated composition, utilizing planning techniques. The descriptions of atomic web services, either semantic (SAWSDL or OWL-S), or not (WSDL), along with user preferences, are used to derive the representation of the web service composition problem as a planning problem, in standard PDDL.[8] Semantic information, acquired from a variety of sources to accommodate both the semantic and non-semantic descriptions cases, is integrated in the standard PDDL notation; therefore, solutions can be acquired by means of utilizing standard planners. Thus, the web service composition problem can be solved as a standard planning problem, taking advantage of existing methods, algorithms and tools, independently of the initial representation standard.[13,19] The solution to this problem constitutes a composite service description, described either in OWL-S or BPEL4WS[3] standards. Semantic awareness is facilitated by acquiring information from ontology analysis and lexical thesauri; reasoning over the hierarchical relationships between concepts reveals semantic equivalences or similarities, which can be exploited through semantic relaxation to provide approximate solutions.[11] An additional issue addressed is composite service quality assessment, in cases where more than one composite services, exact or approximate, satisfy user requirements.

In this paper, the discrete steps of the methodology are identified, independently of their implementation. Moreover, we focus on enhancing non-semantic web service descriptions (WSDL) by extracting semantic information from alternative sources, and embedding this information in PDDL, in order to accommodate seamless composition for both semantic and non-semantic web service descriptions. Alternative semantic relaxation techniques to provide approximate composition solutions and facilitate composite service quality assessment are also explored.

Also, in the paper, the proposed methodology is consequently supported by the development of an extendable integration framework, which incorporates, in a modular fashion, software components accommodating discrete methodology steps. Each step may be implemented using alternative techniques, integrated as discrete framework modules. The structure of this framework in its current version features a modular architecture, handling alternative web service description standards and alternative semantic enhancement and relaxation methods in a unified fashion. It is created utilizing

experience stemming from the development of existing software tools such as the one presented in Ref. 12. The differences of this work compared with previous work performed by the authors are also described in more detail in the next section.

The rest of the paper is organized as follows: Section 2 discusses related work, while Section 3 provides an overview of the proposed methodology. Sections 4 and 5 elaborate on the basic and optional steps, focusing on the semantic aspects of web service descriptions. Section 6 provides an overview of the proposed framework, focusing on implementation issues concerning semantic relaxation. Section 7 presents a case study, where web services described in OWL-S and WSDL are handled uniformly to compose other services using the proposed methodology and framework. Conclusions and future directions reside in Section 8.

2. Related Work

Automated Planning is a well defined and long-studied AI field which has been successfully applied to many areas to automate problem solving. Web service composition is among these areas, as planning can accommodate automation of both the generation of the composition plan and the discovery of the appropriate atomic web services, enabling efficient management of the vast volume of the web services domain, while maintaining scalability, flexibility to detect changes in atomic service definitions, and dynamic handling of service failure/unavailability. The employment of intelligent planning techniques for web service composition can be significantly facilitated by the use of semantics.

Theoretical works, such as the Causal Link Matrix (CLM),[21] provide a solid background for semantic web service composition through AI techniques. CLM constitutes a formal theoretical model accommodating AI planning for web service composition. It involves precomputing all causal relations between semantic web services and utilizing them to formulate valid compositions. Although it takes into account semantics, the lack of an implementation and experimental results does not allow us to draw conclusions about its scalability.

SHOP2[41] was initially created as a general-purpose, heuristic-driven HTN planning system and was later used for automated web service composition. OWL-S process models are encoded as SHOP2 domains, and solutions are acquired by HTN planning. The main disadvantage of this approach is that the planning process, due to its hierarchical nature, requires certain decomposition rules to be encoded in advance with the help of a DAML-S process ontology. In order for decomposition rules to be sound, prior expert knowledge of the domain is required.

Another approach for automated web service composition is attempted through planning as model checking, with the modification of the MBP system.[34] MBP accepts as input web services, described as abstract processes in BPEL4WS, and a given goal process. It produces a description of the desired composite service in BPEL4WS. This approach copes with issues such as non-determinism, partial observability and extended

goals. However, semantic information is not utilized during composition, while scalability is questionable.

The work in Ref. 25 represents atomic services as state transition operators and employs estimated-regression planning with heuristics to perform composition. In order to be used, it requires extension to current standards, while scalability results are not encouraging.

The framework presented in Ref. 29 attempts automated web service composition for semantic web services, by transforming the available web services into Event Calculus axioms and then utilizing abductive planning to formulate a solution. The solution is finally converted to OWL-S to facilitate execution. A main unresolved issue of this framework, and part of future work, is scalability.

The approach presented in Ref. 26 attempts the modification of GOLOG to adjust it to web service composition standards. The approach is based on intelligent agents having the ability to reason for automated service discovery and composition. User requirements and constraints are modeled through Situation Calculus. Consequently, GOLOG is used to find an appropriate composition plan. Encoding and translation processes in this approach are generally complex, while interoperability with existing systems and standards is decreased.

The SWORD system[35] describes available web services with the aid of Entity-Relationship Models and Horn rules. Therefore, domain-specific knowledge is required. The final composition plan is derived through a rule-based expert system, requiring user intervention.

The work in Ref. 9 adopts a declarative approach to dynamically compose SAWSDL web services, using planning. The proposed approach overcomes the issue of syntactic heterogeneities both in the data and the functional level, by performing the matching through mediation, exploiting semantic information present in SAWSDL, and utilizes an extension of the GraphPlan to acquire a solution as a BPEL file.

OWLS-XPlan[19] uses semantic descriptions of web services in OWL-S to derive planning domains and problems, and then invokes a planning module, called XPlan, to generate composite services. The system is compliant with an XML dialect of PDDL. However, semantic information provided from domain ontologies is not utilized; therefore, the planning module requires exact matching between service inputs and outputs.

The PORSCE II system[12,13] was developed by the authors to perform automated semantic web service composition through planning, using OWL-S web service descriptions. The web service composition problem was transformed in planning terms, using the PDDL standard, and enhanced with semantic information extracted from OWL ontologies, enabling the generation of both exact and approximate solutions by external planners. The PORSCE II system provided a solid implementation for OWL-S web service composition, dealing with semantics, representation issues, as well as performance issues.

All aforementioned systems handle the web service composition problem fragmentarily, providing solutions only for a specific description standard at a time, without providing integration for different standards or extensive semantics support. At the same time, previous research concerning web crawling in search of web service descriptions[14] revealed that the largest fraction of web service descriptions that can be found on the web is described in the WSDL standard. The issue with this standard is that, while it is very efficient for development purposes, it does not provide inherent semantic information to enable semantic composition and relaxation. For this reason, novel ways of introducing semantics in this standard must be explored. Moreover, the approach implemented in PORSCE II did not deal with integration issues, in cases when the composition satisfying user needs is comprised of web services described in different standards, for example when some web services are described in WSDL and others in OWL-S. For this, a unifying methodology, unaware of the standards used for describing building components should exist.

Based on these observations, we resorted to developing a methodology encapsulating the approach of the PORSCE II system, and enhancing it in a way that could be applied to all existing, broadly used web service description standards in a uniform fashion that provides additional value, namely resulting in the development of the PORSCE Framework. To provide a solid yet flexible implementation of each step, the underlying components needed to be modular. Each step of the methodology may be applied independently of the underlying web service standards and specific implementations; for example, in order to integrate a new web service standard enabling composition, an additional module can be integrated into the corresponding step. Thus, the proposed integrated methodology for web service composition can be applied independently from the underlying tools.

Taking into account the functionality of the systems presented in this Section, as well as through the experience gained from the development of the PORSCE II system, we concluded the following concerns that had to be taken into account when designing the PORSCE Framework:

(a) The architecture of such environments[41,9,19,12,13] should be modular, enabling the decomposition of the supported functionality into independent modules implemented by autonomous software systems, such as planners, PDDL generators for discrete web service description languages, semantic manipulation algorithms, etc., communicating through PDDL and web service standard descriptions.
(b) Alternative web service description languages, either semantically enriched or not, should be supported and handled in a uniform way, so that the user can experience seamless and consistent composition.
(c) Alternative semantic enhancement and relaxation methods could be applied.

Summarizing, in this paper, extending our previous work in Refs. 12 and 13, we propose a well-defined and generic methodology to address the web service composition problem utilizing semantics, for both semantic and non-semantic web service

descriptions. The approach is modular, structured in discrete steps and independent from implementation aspects, such as specific planning or semantic manipulation algorithms. In this way, its implementation and the acquisition of solutions can be facilitated through the integration of a number of different systems, handled as discrete modules of an extendable framework.

3. Methodology Overview

The proposed methodology aims at offering a unified approach for both semantic and non-semantic descriptions of web services, enabling the combination of services in the same composite web service plan, even when they are described in different standards, such as WSDL, SAWSDL and OWL-S. It also takes into account semantics using different techniques, such as ontologies, when available, or other external sources.

To achieve automated web service composition, the following steps are required:

(i) Definition of the web service composition problem. This includes determination of the requirements concerning the desired composite service as well as discovery of the available atomic or composite web services that may act as building blocks for the required composite service.
(ii) Transformation of the web service composition problem to a planning problem. This includes the production of a planning domain and the corresponding problem and representing it using a standard planning language, such as PDDL.
(iii) Solving the planning problem via planning algorithms.
(iv) Expressing the solution in web service context.

The composition process performed through the aforementioned steps is able to provide composite services built from a set of available atomic and composite ones, according to user requirements. The provided solutions can be significantly improved by infusing semantics, based on the following steps:

- Semantic enhancement of the produced planning domain and problem with semantically similar or equivalent concepts. Semantic information can be obtained either directly, by semantic web service descriptions, or indirectly, by combining web service descriptions and other sources such as thesauri.
- Semantic relaxation of the produced planning problem, which requires prior semantic enhancement, and permits the formulation of approximate solutions.
- Quality and accuracy assessment of the produced composite services, in case semantic relaxation produces approximate solutions.

The proposed methodology is depicted in Figure 1.

Each step is designed to function independently and only exchange of data takes place between them, using the OWL-S, (SA)WSDL and PDDL standards. As a result, each of the methodology steps can be implemented by a different software system.

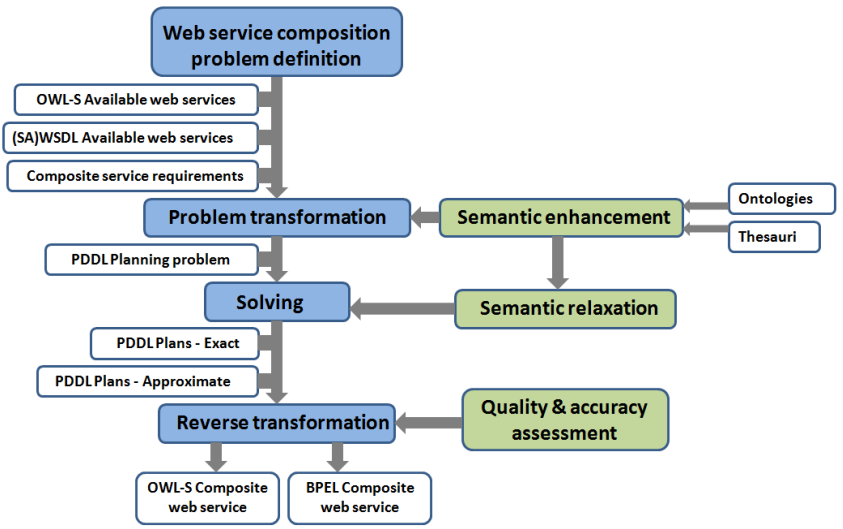

Fig. 1. Proposed methodology steps.

4. Basic Steps

4.1. *Definition of the web service composition problem*

When in need for a web service that performs a certain task or has some specified functionality, the user does not know a priori if their request will be satisfied by an atomic or a composite service. The need for composite services occurs from the inadequacy of simple services to satisfy complex user needs, or reflect intricate business processes. The definition of the web service composition problem includes two elements:

(a) available atomic or composite web services that act as building blocks
(b) user requirements concerning the desired composite service, in terms of inputs and outputs.

The available web services can be found by UDDI registries. However, many UDDI registries are focused on specific areas and do not provide an easy method for accessing the descriptions they contain. A more promising but time-consuming alternative is crawling.[14] Web-crawlers can collect web service descriptions by wandering around the web in a regular fashion while indexing only specific file types.

4.2. *Transformation*

Consequently, the web service composition problem must be transformed and represented in planning terms.

A planning problem is usually modeled according to STRIPS notation[6] as a tuple $\langle I, A, G\rangle$ where I is the initial state, A a set of available actions and G a set of goals. States are represented as sets of atomic facts. Set A contains all the actions that can be used to

modify states. Each action A_i has three lists of facts containing the preconditions of A_i, the facts that are added to the state and the facts that are deleted from the state, noted as $prec(A_i)$, $add(A_i)$ and $del(A_i)$ respectively. An action A_i is applicable to a state S if $prec(A_i) \subseteq S$, yielding the successor state S', calculated as $S' = S - del(A_i) \cup add(A_i)$. The solution to a planning problem (plan P) is a sequence of actions $P = \{A_1, A_2, \ldots, A_n\}$, which, if applied to I, lead to a state S' such that $S' \supseteq G$.

A straightforward solution for mapping the web service composition problem to a planning problem is the following: The set IC of concepts that the user provides as inputs to the composite service formulate the initial state, while the desired outputs GC of the composite service formulate the goals of the problem: $I = IC$ and $G = GC$. The set of available actions A occurs by translating each available web service description WSD_i into a domain action A_i.

For the OWL-S web service descriptions case, the action set formulation is described in Ref. 13.

For non-semantic web service descriptions in WSDL and semantic descriptions in SAWSDL, the actions are presented is this paper and formulated as following:

- The name of the action is the *name* attribute of the *service* element of the description:

$$name(A_i) \equiv WSD_i.service : name$$

- The preconditions of the action are formed based on the *input* element of the *operation* structure of the description:

$$prec(A_i) \equiv \bigcup_{k=1}^{n} \{WSD_i.operationinput_k : element\}$$

- The add effects of the action are formed based on the *output* element of the *operation* structure of the description:

$$add(A_i) \equiv \bigcup_{k=1}^{n} \{WSD_i.operationoutput_k : element\}$$

- The delete list is left empty, since WSDL does not have a way to express negative results of a web service:

$$del(A_i) \equiv \varnothing$$

For semantic web service descriptions in OWL-S, the Service Profile instance is used:

- The name of the action is the *rdf:ID* field of the profile:

$$name(A_i) \equiv WSD_i.ID$$

- The preconditions of the action are formed by the service input and precondition definitions:

$$prec(A_i) \equiv \bigcup_{k=1}^{n} \{WSD_i.hasInput_k\} \cup \bigcup_{k=1}^{m} \{WSD_i.hasPrecondition_k\}$$

- The add effects of the action comprise of the service output and positive effect definitions, while the delete list is formed by the negative effect definitions:

$$add(A_i) \equiv \bigcup_{k=1}^{n} \{WSD_i.hasOutput_k\} \cup \bigcup_{k=1}^{m} \{WSD_i.hasEffect_k^+\}$$

$$del(A_i) \equiv \bigcup_{k=1}^{m} \{WSD_i.hasEffect_k^-\}$$

4.3. *Solving via planning systems*

The produced sets I, G and A must be encoded in a standard planning language; the recommended one is PDDL, since it is the prominent standard input language for the majority of existing planners.[7,18] Encoding in PDDL ensures independence between transformation and solving steps of the proposed methodology. Any PDDL-compliant external planning system can be used to implement this step, ensuring that contemporary, improved planners can be easily integrated.

4.4. *Expressing the solution in web service context*

The acquired solutions have to undergo a reverse translation process to be expressed in a web service standard, for accommodating composite service deployment and execution monitoring. Such standards are BPEL4WS, for the non-semantic case, which can be executed in engines such as IBM WebSphere Business Integration Server Foundation[45] or ActiveBPEL (ActiveBPEL), and OWL-S for the semantic case, which can be executed in engines such as OWL-S Virtual Machine.[33]

The reverse translation algorithms for OWL-S have been presented in Ref. 12; reverse translation process for the WSDL case is presented hereby.

Algorithm 1 presents the basic algorithm that creates a composite service, given a web service graph, by identifying processes that can be executed parallel or in sequence.

Algorithm 1 Computes an initial composite service with *sequence* and *flow_start* constructs
Inputs $G = (V, E)$: the web service graph
Output C: a composite service with *sequence* and *flow_start* constructs

```
1     set R ← {r ∈ V : ∀x ∈ V, (x → r) ∉ E} // R is the set of root nodes in G
2     if |R| = 0 then return NULL
3     if |R| = 1 then
4       set G′ ← the tree in G with r ∈ R as the root
5       return sequence(r,Basic(G′ − {r}))
6   set c ← { }
7       for each r in R
8         set G′ ← the tree in G with r ∈ R as the root
9          set c ← c ∪ Basic(G′ − {r})
10    return flow_start(c)
```

Algorithm 2 (Join) Replaces flow_start with flow where possible in a composite service
Inputs C = f(a1, a2,…, an): a composite service with sequence and flow_start constructs
Output C: a composite service with sequence and flow constructs

set f(a_1, a_2, .., a_n) = C, where *f* is the name of the construct and a_1 to a_n its arguments
  if f = NULL then return NULL
  if f = sequence then
  a'_1 = flow_end(a_1)
  a'_2 = flow_end(a_2)
  return f(a'_1, a'_2)
 if f = flow_start then
  for each pair (a_i, a_j), i, j in [1,n]
if a_i and a_j have a common ending, i.e. $a_i = a'_i \cup k$ and $a_j = a'_j \cup k$
  then $C' = C - \{a_i, a_j\} \cup$ seq(flow(a'_i, a'_j), k)
  return C′

A web service graph is a graph $G = (V, E)$, where the nodes in V correspond to all the atomic services in the plan and the edges $(x \rightarrow y)$ in E, where x and y are nodes in V, define that web service x produces an output that is required by y as an input. Algorithm 1 processes every root node in the graph and produces as output a composite construct of either the form *sequence*(c_1, c_2), or *flow_start*(c_1, c_2,…, c_n), where c_1 to c_n are either *NULL* or composite constructs.

Algorithm 1 only identifies the starting point of a parallel flow; therefore, its output must consequently be fed to Algorithm 2, which searches all possible pairs of parallel flows, in order to find a common ending part, where the processes must be executed in sequence again. At the end of this algorithm, all parallel parts of the process are enclosed in *flow* constructs and all sequential parts are indicated by *sequence* constructs; these can be directly encoded using the homonym BPEL4WS elements.

5. Semantics Infusion Steps

5.1. *Semantic analysis and enhancement*

The semantic analysis step provides semantic information concerning equivalent and semantically similar ontology concepts to a given query concept, allowing the realization of semantic relaxation in order to acquire approximate solutions, in cases where exact input/output matching is not available.

Intuitively, two concepts are considered semantically relevant by the semantic analysis module if and only if

(a) they have a specific semantic relationship (including semantic equivalence), and
(b) their semantic similarity, in terms of a specific semantic distance/similarity measure, exceeds a user-defined threshold, allowing the adjustment of the concept relevance criterion, enabling the incorporation of different degrees of relaxation.

Formally, the definitions for concept relevance are provided in the remainder of this paragraph.

Let O denote the set of the available ontology concepts, F denote the set of the selected hierarchical relations and $a \in [0\ldots1]$ the concept distance threshold, which restricts the distance of two concepts, defining the minimum similarity that is acceptable in order for the concepts to be matched.

Definition 1. A concept $C \in O$ is considered relevant to a concept $D \in O$ with respect to a hierarchical relation set F and a concept similarity threshold a, denoted as $C \approx_a^F D$, if they satisfy at least one hierarchical relation in F and the threshold a on their similarity.

Definition 2. For each concept $C \in O$ its concept relevance set, denoted as R_C, is defined as the set of all the relevant concepts of C, that is, $R_C \equiv \left\{T \in O : T \approx_a^F C\right\}$.

Definition 3. For each set A of concepts, its extended set, denoted as EX_A, is defined as the union of the concept relevance sets of its concepts, that is,

$$EX_A = \bigcup_{\forall C \in A} R_C$$

Definition 4. Two concept sets A and B are relevant, denoted as $A \approx_a^F B$, if all the concepts of one set have at least one relevant concept in the other set and the two sets have the same size, that is,

$$A \approx_a^F B \Rightarrow \forall C \in A, \exists D \in B : C \approx_a^F D \wedge \forall D \in B, \exists C \in A : D \approx_a^F C \wedge |A| = |B|$$

5.1.1. *OWL-S semantic analysis background*

The semantic analysis background for OWL-S ontology-based semantics has been presented in Ref. 12; however, it is briefly summarized in this section as well, in order to have a foundation for the presentation of the semantic analysis for WSDL, which follows in the next section. It should be pointed out that the same semantic analysis performed for OWL-S also holds for the SAWSDL standard, as its semantics are also ontology-based. In order to be able to integrate all three web service description standards in the proposed approach, in all cases, the semantic analysis must be performed in a uniform way and hierarchical relationships as well as semantic distance metrics have to be defined.

5.1.1.1. Hierarchical relationships

The possible hierarchical relationships between two ontology concepts A and B are the following:

- **exact(A, B)**: A and B have the same URI or they are equivalent
- **plugin(A, B)**: A is subsumed by B
- **subsume(A, B)**: A subsumes B
- **sibling(A, B)**: A and B only have a common superclass C.

5.1.1.2. Concept similarity

The semantic similarity between two concepts in the OWL-S and SAWSDL cases is measured as the complement of the semantic distance between these concepts: $Sim(s_1, s_2) = 1 - d(s_1, s_2)$. In the following, two different semantic distance measures are considered.

The *Edge-Counting Distance* (*ec*) computes the distance of two concepts in terms of number of edges found on the shortest path between them in the ontology hierarchy. An edge exists between two concepts *A* and *B* if A is the direct subclass of B, denoted as $A \sqsubseteq_d B$. The ec distance between two concepts considers the following cases:

- $exact(A,B) \rightarrow d_{ec}(A, B) = 0$
- $A \int B \sqsubseteq \perp \rightarrow d_{ec}(A, B) = 1$
- $plugin(A, B) \vee subsume(A, B) \rightarrow d_{ec}(A, B) = p/p_{max}$: where p is the number of edges that exist in the shortest path between A and B in the ontology tree and $p_{\max}$ the maximum *ec* distance found in the ontology, for normalization purposes
- $sibling(A, B) \rightarrow d_{ec}(A, B) = min[d_{ec}(A, T) + d_{ec}(B, T)]$ where T is the least common ancestor.

The *Upwards Cotopic Distance* (*uc*) between two concepts,[24] denoted as $d_{uc}(A, B)$, is defined in terms of the upwards cotopic measure $uc(A)$ that represents the set of the superclasses of the concept A in a hierarchy, including A itself. In the proposed methodology, the upwards cotopic distance definition has been adapted so that similarities that are based on sibling relationships with the generic owl:Thing concept are not considered, and the concept set multiplicity is ignored:

$$d_{uc}(A,B) = 1 - \frac{\left|uc(A) \cap uc(B)\right| - 1}{\left|uc(A) \cup uc(B)\right| - 1}$$

If two concepts are disjoint, their cotopic distance is $d_{uc}(A, B) = 1$; else $d_{uc}(A, B) = v \in [0\ldots1)$.

5.1.2. *WSDL semantic analysis background*

In analogy with semantic analysis for OWL-S and SAWSDL descriptions, this paper introduces semantic analysis for WSDL web service descriptions. WSDL descriptions, unlike OWL-S, do not contain inherent semantic information attached to the concepts used to describe web service inputs or outputs; therefore, semantic information must be acquired by other sources, such as thesauri or lexical databases. One of the largest and most commonly used lexical thesauri is WordNet.[28]

WordNet groups words into sets of semantically or lexically related words, called synsets. This grouping can support the identification of semantically equivalent or related concepts that can be utilized in the proposed approach for semantic awareness and semantic relaxation. Each synset contains a group of concepts that are considered semantically equivalent and is connected to other synsets via hierarchical relationships.

5.1.2.1. Hierarchical relationships

For nouns, which is the case with concepts used as web service inputs and outputs, WordNet semantic relations include:

- **hypernym**: Y is a hypernym of X if every X is a (kind of) Y (canine is a hypernym of dog)
- **hyponym**: Y is a hyponym of X if every Y is a (kind of) X (dog is a hyponym of canine)
- **coordinate term**: Y is a coordinate term of X if X and Y share a hypernym (wolf is a coordinate term of dog, and vise versa)
- **holonym**: Y is a holonym of X if X is a part of Y (building is a holonym of window)
- **meronym**: Y is a meronym of X if Y is a part of X (window is a meronym of building)

These relations form a word taxonomy within WordNet, over which a variety of semantic similarity measures can be applied in order to calculate the semantic similarity of any two concepts.

5.1.2.2. Semantic relevance

A variety of measures for computing the semantic similarity between two concepts can be applied,[4,16,44] which are usually based on the length of the path connecting them, or the identification and manipulation of their common ancestors.

The most representative measure for the first case,[20] calculates semantic similarity between two concepts s_1 and s_2 taking into account the number of nodes in the path, and the maximum depth of the taxonomy:

$$Sim(s_1, s_2) = -\log \frac{length}{2 \cdot D}$$

where *length* is the length of the shortest path connecting s_1 and s_2 and D is the maximum depth of the taxonomy used.

As far as measures involving common ancestors of the two concepts are concerned, most of them incorporate the *Information Content* of the deepest concept that subsumes both concepts; that is, the *least common subsumer*. The *Information Content* of a concept s_0 is

$$IC(s_0) = -\log P(s_0)$$

where $P(s_0)$ is the probability of occurrence of the concept s_0 in a large corpus.

Such measures include Ref. 17 which computes the semantic similarity between s_1 and s_2 as

$$Sim(s_1, s_2) = \frac{1}{IC(s_1) + IC(s_2) - 2 \cdot IC(s_0)}$$

and Ref. 22 which computes similarity as

$$Sim(s_1, s_2) = \frac{2 \cdot IC(s_0)}{IC(s_1) + IC(s_2)}.$$

For the proposed approach, we decided to incorporate a sophisticated measure of semantic relatedness, called *Omiotis*,[43] which captures relatedness in multiple granularity levels, for example between two concepts (words), as well as between groups of words; therefore, it can be used not only for semantic awareness and relaxation purposes but also for composite service accuracy assessment. The *Omiotis* measure calculates semantic relatedness by utilizing the semantic network that can be constructed by taking into account all semantic relations between concepts. It considers the path length, captured by compactness, and the path depth, captured by semantic path elaboration. Semantic relatedness between two groups of words A and B is calculated as

$$Omiotis(A, B) = \frac{\left|\zeta(A, B) + \zeta(B, A)\right|}{2},$$

where

$$\zeta(A, B) = \frac{1}{|A|}\left(\sum_{a \in A} \lambda_{a,b_*} \cdot SR(a, b_*)\right), \quad a_* = \arg\max_{a \in A}(\lambda_{a,b} \cdot SR(a, b)) \quad \text{and}$$

$$b_* = \arg\max_{b \in B}(\lambda_{a,b} \cdot SR(a, b))$$

The lexical relevance $\lambda_{a,b}$ between terms $a \in A$ and $b \in B$ is calculated as the harmonic mean of the respective terms' TF-IDF values, as determined by the standard TF-IDF weighting scheme:[38]

$$\lambda_{a,b} = \frac{2 \cdot TFIDF(a, A) \cdot TFIDF(b, B)}{TFIDF(a, A) + TFIDF(b, B)}$$

5.2. *Semantic relaxation*

If a planning system, due to information provided by the step of semantic analysis, is aware of semantic equivalence among syntactically different concepts, it is able to match them during planning.[32] Furthermore, in cases where no exact matching of concepts is possible, the semantic analysis step is able to provide not only equivalent concepts but also semantically similar ones as well. In this case, semantic relaxation takes place and approximate matching can be performed, leading to the formulation of less accurate composite services.

The important aspect that needs to be clarified in this paper, concerning integration and utilization of all web service description standards, semantic and non-semantic, in a uniform way, is that the semantic relaxation step, designed using the proposed methodology, is independent from the previous semantic analysis step. As a consequence, the existing methodology step that includes all the required actions in order to semantically

match web service inputs and outputs can remain unchanged for the WSDL case, as well as the integration of all standards, since it is unaware of the semantic context.

The existing approach for semantic relaxation has been extensively presented and analyzed in Ref. 12; hereby only a short overview will be presented for completeness purposes.

Semantic relaxation is performed by enhancing the problem at hand with all relevant concepts for both facts of the initial state and outputs of the available actions, according to Definition 1. The original problem $\langle I, A, G\rangle$, is semantically relaxed, resulting to the enhanced problem $\langle EIS, EAS, EGS\rangle$, based on the following rules:

- The original set of concepts in the initial state I is replaced by its extended set (*Extended Initial State — EIS*), according to Definition 3.
- The effects list of each action is replaced by its extended set according to Definition 3, producing the *Extended Action Set* (*EAS*).
- The goals of the problem remain the same, since extensions of the initial state and the action set enable approximate matching.

The semantically enhanced problem, as far as planning systems are concerned, is no different than a classical planning problem; therefore, semantic relaxation is independent from the solution step of the methodology.

5.3. *Composite service assessment*

In many cases, multiple composite services are produced, due to semantic relaxation and the use of different planners; therefore, the ability to assess them is important. Currently, included assessment metrics concern statistics, such as the number of actions and the number of levels in a plan, and a distance quality metric, which indicates the accuracy of the plan.

If no semantic relaxation is performed then plans with fewer levels are preferable, as their execution is estimated to take less time; among plans with the same number of levels, the ones consisting of the fewer actions are preferable. When semantic relaxation takes place, the distance quality metric gains particular significance, as accurate plans are generally more preferable than approximate ones, even if they involve more actions.

In order to calculate the distance quality metric, each concept appearing in the plan is annotated with a semantic distance d_i with respect to the original concept it was derived from and the selected similarity metric. Additionally, each concept is annotated with the kind of hierarchical relationship to the original concept, corresponding to a weight w_i. If there are a total of n concepts, the plan similarity metric is calculated as:

$$PSM = \prod_{i=0}^{n} w_i \cdot d_i$$

If there is an exact input-to-output matching, or only equivalent concepts are used, then plan accuracy quality is 1, decreasing as the plan becomes less accurate.

In order for the composite service to be assessed in a uniform way, even if it includes primitive services expressed in different standards and thus semantically relaxed using a different metrics, all distances should be normalized to [0…1].

6. Implementation Framework

The implementation of the aforementioned methodology is accommodated by the *PORSCE framework for automated web service composition through planning*.

The PORSCE framework is modular and includes the following subcomponents: Parsers, Transformation Component, Reverse Transformation Component, Semantic Analysis Component, Visualizer and Service Replacement Component. Each component can be implemented using different algorithms, and can be connected to the framework in a plug-in fashion.

The Parser parses available web service descriptions. It comprises of two different components for OWL-S web services and WSDL/SAWSDL web services.

The Semantic Analysis Component discovers semantically similar concepts, and it includes modules that take into account domain ontologies, to serve OWL-S and SAWSDL standards, as well as a module for utilizing existing systems for semantic similarity such as *Omiotis*,[43] as well as WordNet, to deduce semantic information. Current version of the semantic analysis and relaxation component for the WSDL case can be found at http://galaxy.hua.gr/~raniah/files/semanticSimZap.zip.

Transformation Component translates the web service composition to a planning problem and semantically enhances it. Additionally, it invokes external planners (currently LPT-td and JPlan), which produce solutions and assesses the accuracy of the approximate ones. The Reverse Transformation Component expresses the produced results in web service terms. The Visualizer provides a visual description of the composite service. Finally, Service Replacement Component enables the replacement of a specific atomic web service. The architecture of the PORSCE framework is depicted in Figure 2.

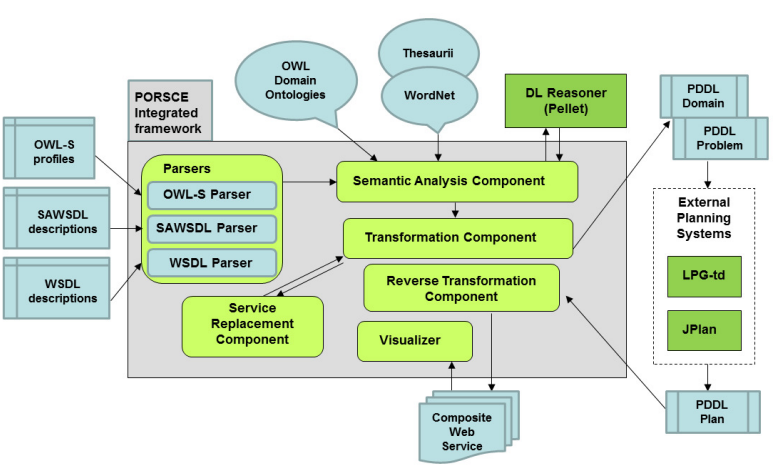

Fig. 2. Integrated architecture of the PORSCE framework.

The modular design of the PORSCE framework facilitates the substitution of modules as plug-ins, in order to experiment with different configurations

The prototype implementation of the framework[10] incorporated the OWL-S Parser and the corresponding Transformation Component, as well as a Semantic Analysis Component, enabling semantic relaxation following Algorithm 3. Quality assessment of the approximate solutions was performed by using only the edge-counting semantic distance metric. *PORSCE* acquired solutions to the web service composition problem by an external local planner, JPlan (JPlan).

Conclusions drawn from the prototype lead to the development of the second alternative implementation of the framework, *PORSCE II*.[11] The semantic analysis component was enriched with an additional module, implementing Algorithm 4. Furthermore, the cotopic semantic similarity metric was introduced, offering flexibility in defining concept similarity. The service replacement component was also implemented, handling cases of service failure and a visual interface was included. In order to highlight the independence between the problem representation and planning systems, an additional external planner was added, LPG-td.[7]

Currently, the third version of the framework, *PORSCE III* incorporates components implementing all steps described in the proposed methodology. It also implements the aforementioned techniques for semantic enhancement and relaxation. An important advantage of the *PORSCE III* system is the ability to uniformly handle domains including web service descriptions in OWL-S or SAWSDL (under the assumption that semantic annotations are expressed in OWL), and WSDL.

7. Case Study

This section presents a case study from the e-government domain, in order to demonstrate the application and functionality of the proposed methodology. The scenario concerns the student registration process to a University Department, at the beginning of their studies. In the real-world case in Greece, the person needs to collect:

- an admission certificate, issued from the Ministry of Education, certifying that the student successfully participated in the national admission examinations and admitted to this specific Department
- a family status certificate, issued by the Municipality the student belongs to, confirming specific housing or food privileges, they are entitled to
- a health certificate, issued by the Department of Public Health, which certifies that all necessary physical examinations were conducted

When registering, the student submits the certificates, along with proper identification and recent photographs to the Department Secretariat. The registration can be used by the student to obtain a public transport pass, a library card and an electronic account from the University Network Operation Centre, enabling them to use all the electronic services offered by the University and the Ministry of Education. An example of such services is

Eudoxus, a service for selecting, registering and distributing teaching material and academic books.

For the implementation and integration of these e-government services, the following atomic and autonomous web services should be provided from the corresponding authorities:

- *Registration WS*: A web service offered by the Department Secretariat. It accepts as inputs the certificates and a photograph of the student, and performs the registration
- *Library Registration WS*: A web service offered by the University Library. It accepts as input the student registration number and issues the student library card
- *Create Account WS*: A web service offered by the University Network Operation Center. It accepts as input the registration and creates a student account (username & password).
- *Student Pass WS*: A web service offered by the Public Transport Organization. It accepts as inputs the student registration number and a photograph and issues the student public transport pass.
- *Admission WS*: A web service offered by the Ministry of Education. It accepts as inputs the student's name, surname and ID and issues as output an admission certificate, indicating the Department the student may be admitted to.
- *Health WS*: A web service offered by the Regional Health Department. It accepts as inputs the student's name, surname and ID and issues as output a health certificate, ensuring that the student has undertaken all required examinations.
- *Family Status WS*: A web service offered by the Municipality the student was born in. It accepts as inputs the student's name, surname and ID and issues as output a family status certificate, indicating certain family features that are taken into account to define student privileges.
- *Eudoxus WS*: A web service offered by the Ministry of Education. It accepts as input the student account and creates a Eudoxus account, enabling the student to register and obtain academic books and teaching material.

For each of the aforementioned web services, there might be several alternatives, offered from different authorities; the specific names of each alternative web service depend on the providing authority. Note that citizens (users) are identified by each authority using independent authorization techniques. As far as description standards are concerned, in this case OWL-S descriptions are provided for the web services offered by University authorities, while the rest web services are described in WSDL. Most of these web services were developed for the needs of the research presented in Ref. 15; current version of this project, which includes most of the wrappers for these web services, can be found at https://github.com/meletakis/collato, under the working name CollaTo (Collaboration Tool) project. For the case of alternative similar web services to the ones required, or for the case when external web services are unavailable from the

corresponding authorities, in order to check the functionality of the proposed platform with different plans, additional web services were developed as localhost simulated services. In order to enhance the web service domain with numerous web service descriptions, web services retrieved from Refs. 31 and 14 were also considered. OWLS-TC also included the ontology used in this example, namely *egov.owl*, which concerns e-government and e-administration concepts.

Note that the case study presented hereby, as well as all cases that the proposed framework targets, are highly dynamic. Firstly, user requirements for composite services change over time; even for completing the same task, different users have different needs. In addition, the environment in which the framework operates is highly dynamic as new services are added, some of the old services might become unavailable, and service descriptions may be altered.

When using PORSCE for accommodating web service composition, an essential step is selecting the available inputs and the desired outputs of the composed service. In the University registration use case, the composite service should perform the student registration and also issue all related documents, e.g. the Registration Certificate and the Student ID card. The available inputs are the student's name and surname, his ID and a photograph. The desired outputs are the registration at the selected Department, the public transport pass, the library card, the NOC account and the Eudoxus account. The dialog for facilitating graphical selection of inputs and outputs is depicted in Figure 3. Note that the set of available inputs (or outputs) is produced as the union of all sets of inputs (or outputs) of all available web services. This comprises the Web Service Composition Problem Definition Step of the proposed methodology.

The available inputs are mapped to the initial state of the planning problem, while the desired outputs are mapped to the goal state. The available web services are represented as operators in the planning domain, comprising the Problem Transformation Step. The formulated planning domain and problem is then encoded into PDDL and forwarded to external planning systems that produce a solution, which is visualized in Figure 4 (Solving & Reverse Transformation Steps).

No semantic relaxation is required, as the inputs and outputs matched. However, this is hardly the case in the real world, where web services are implemented by many different providers. As an example, consider the case where the Admission WS describes

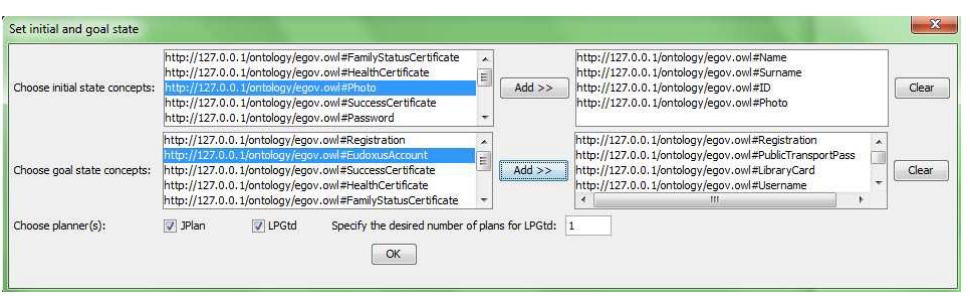

Fig. 3. Dialog for selection of composite web service available inputs and desired outputs.

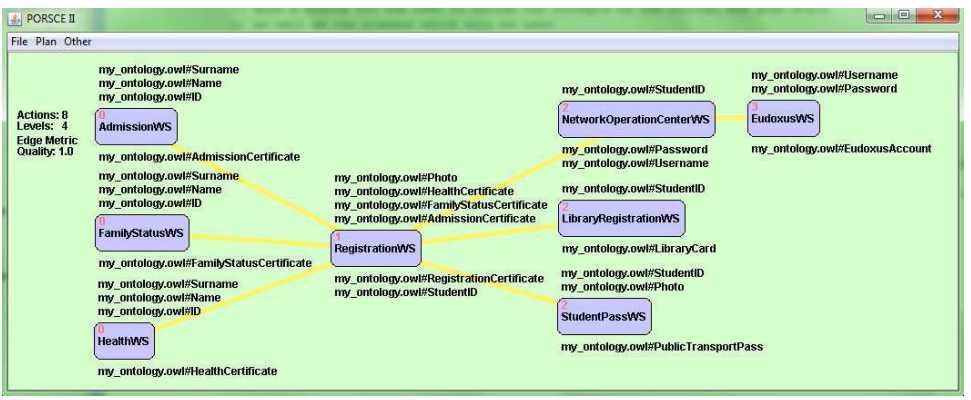

Fig. 4. Produced composite web service plan (exact matching).

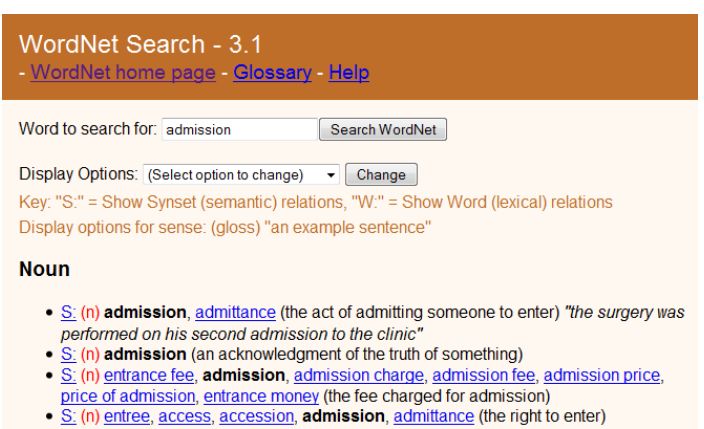

Fig. 5. Semantic similarity of the words "admission" and "admittance" in WordNet.

its output as "AdmittanceCertificate", while the Registration WS describes its corresponding input as "AdmissionCertificate". Exact matching in this case would not produce a solution; however, semantic relaxation using WordNet would conclude that the words "admission" and "admittance" are semantically similar (Figure 5); therefore, the PORSCE framework would be able to provide an approximate solution based on this semantic information (Figure 6) (Semantic Enhancement and Semantic Relaxation Steps).

If the available web services were always specific and unaltered throughout their lifecycle, and the student requirements were in all cases, there would not be essential need for dynamic, automatic web service composition. Instead, a simple BPEL definition for the composite service, describing the way the available services should interact to achieve the required functionality, would suffice. However, this is not the case here, as each student may have different requirements; for example, a certain student might not wish to use the benefits of a family certificate, or another student might not wish a library card.

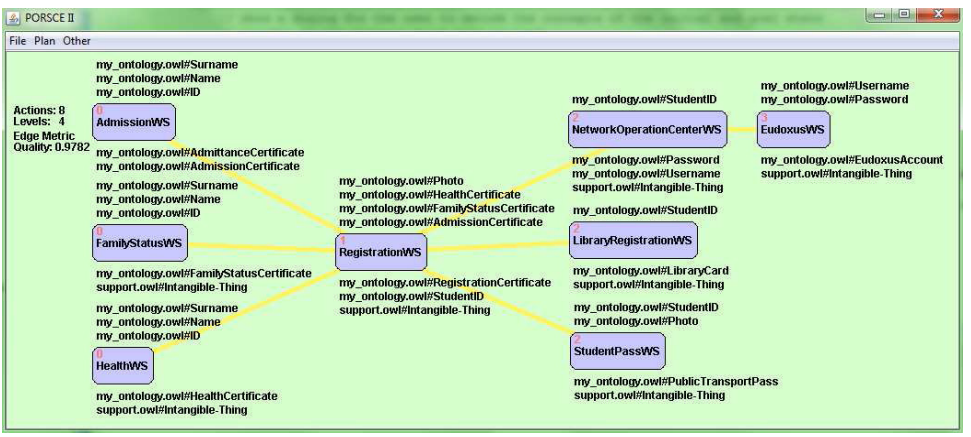

Fig. 6. Produced composite web service plan (approximate matching).

Furthermore, there might be alternative compositions that accomplish the required functionality, as some specific functionality could be offered as a web service by more than one providers, or user requirements could be satisfied by structurally different combinations of atomic web services. Alternative compositions are evaluated in the Quality & Accuracy Assessment Step.

Both these factors affect the desired composite service plan; if the issue of composing the available web services were to be handled through static BPEL composite services, the designer would have to create and provide all meaningful combinations manually.

8. Conclusions and Future Work

The work presented in this paper concerns a generic, unified and structured methodology for automated web service composition, utilizing AI planning techniques.

The first step of the proposed methodology concerns determination of the requirements concerning the desired composite service as well as discovery of the available web services, resulting in the definition of the web service composition problem. The web service composition problem is consequently transformed into a planning domain and corresponding problem, represented in a standard language and solved via planning. Finally, the solution is transformed back to web service context. The solutions provided by the basic steps are further elaborated and improved by additional processes, which result to semantic enhancement and relaxation.

The overall methodology consists of independent steps and each can be accommodated by one of more software subsystems which communicate through standard languages. During implementation, modularity allowed a variety of algorithms to be incorporated. The resulting framework is capable of handling both semantic (OWL-S, SAWSDL) and non-semantic (WSDL) web service descriptions, and incorporating semantic information from different sources, such as ontologies and thesauri.

The proposed methodology was supported and tested by the development of the corresponding *PORSCE* framework. Performance measurements obtained from experiments indicate the scalability of the framework for large web service domains.

Among the main challenges that the methodology should address in the future is the incorporation of non-functional requirements during web service composition, such as quality of service or availability. Additional modules should be added to the PORSCE framework.

References

1. ActiveBPEL engine, http://www.activebpel.org/ [retrieved 12/05/13].
2. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, Web Services Architecture, W3C Working Group Note 11 (2004) 2005–1.
3. BPEL4WS, Business process execution language for web services version 1.1, 2005, http://www.ibm.com/developerworks/library/specification/ws-bpel/ [retrieved 12/05/13].
4. A. Budanitsky and G. Hirst, Evaluating wordnet-based measures of lexical semantic relatedness, *Computational Linguistics* **32** (2006) 13–47.
5. M. Carman, L. Serafini and P. Traverso, Web service composition as planning, in *ICAPS 2003 Workshop on Planning for Web Services* (2003).
6. R. E. Fikes and N. J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, in *IJCAI'71*: *Proc. of the 2nd Int. Joint Conference on Artificial Intelligence* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1971), pp. 608–620.
7. A. Gerevini, A. Saetti, I. Serina and P. Toninelli, LPG-TD: A fully automated planner for PDDL2.2 domains, in *Proc. of the 14th Int. Conf. on Automated Planning and Scheduling* (*ICAPS-04*) (International Planning Competition, 2004).
8. M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld and D. Wilkins, PDDL — The planning domain definition language, Tech. rep., Yale University, New Haven, CT (1998).
9. K. Gomadam, A. Ranabahu, Z. Wu, A. P. Sheth and J. A. Miller, A declarative approach using SAWSDL and semantic templates towards process mediation, *Semantic WS Challenge*, Semantic Web and Beyond, Vol. 9 (2008), pp. 101–118.
10. O. Hatzi, G. Meditskos, D. Vrakas, N. Bassiliades, D. Anagnostopoulos and I. Vlahavas, A synergy of planning and ontology concept ranking for semantic web service composition, in *IBERAMIA '08: Proc. of the 11th Ibero-American Conf. on AI* (Springer-Verlag, Berlin, Heidelberg, 2008), pp. 42–51.
11. O. Hatzi, G. Meditskos, D. Vrakas, N. Bassiliades, D. Anagnostopoulos and I. Vlahavas, Semantic web service composition using planning and ontology concept relevance, in *WI-IAT '09: Proc. of the 2009 IEEE/WIC/ACM Int. Joint Conf. on Web Intelligence and Intelligent Agent Technology* (IEEE Computer Society, Washington, DC, USA, 2009), pp. 418–421.
12. O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos and I. Vlahavas, The PORSCE II framework: Using AI planning for automated semantic web service composition, *The Knowledge Engineering Review* (Cambridge Press, 2011).
13. O. Hatzi, D. Vrakas, M. Nikolaidou, N. Bassiliades, D. Anagnostopoulos and I. Vlahavas, An integrated approach to automated semantic web service composition through planning, *IEEE Transactions on Services Computing* **5**(3) (2012a).
14. O. Hatzi, G. Batistatos, M. Nikolaidou and D. Anagnostopoulos, A specialized search engine for web service discovery, *IEEE Int. Conf. on Web Services* (*ICWS 2012*) (Honolulu, Hawaii, USA, June 2012b).

15. O. Hatzi, M. Nikolaidou, P. Katsivelis, V. Hudhra and D. Anagnostopoulos, Using social network technology to provide e-administration services as collaborative tasks, *International Conference on Electronic Government and the Information Systems Perspective and International Conference on Electronic Democracy* (*EGOVIS/EDEM 2012*) (Vienna, Austria, 2012c).
16. A. Hliaoutakis, G. Varelas, E. Voutsakis, E. G. M. Petrakis and E. Milios, Information retrieval by semantic similarity, *International Journal on Semantic Web and Information Systems* **2** (2006), pp. 55–73.
17. J. J. Jiang and D. W. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, *Int. Conf. Research on Computational Linguistics* (*ROCLING X*) (1997).
18. JPlan, Java Graphplan Implementation, http://sourceforge.net/projects/jplan [retrieved 27/10/10].
19. M. Klusch and A. Gerber, Semantic web service composition planning with OWLS-XPlan, in *Proc. of the 1st Int. AAAI Fall Symp. on Agents and the Semantic Web* (2005), pp. 55–62.
20. C. Leacock, M. Chodorow and G. A. Miller, Using corpus statistics and WordNet relations for sense identification, *Computational Linguistics* **24**(1) (1998) 147–165.
21. F. Lecue and A. Leger, A formal model for semantic web service composition, in *Int. Semantic Web Conf.* (2006), pp. 385–398.
22. D. Lin, An information-theoretic definition of similarity, in *Proc. of the 15th Int. Conf. on Machine Learning* (Morgan Kaufmann, 1998), pp. 296–304.
23. J. Lu, D. Ruan and G. Zhang, E-service intelligence: An introduction, in *E-Service Intelligence* (2007), pp. 1–33.
24. A. Maedche and V. Zacharias, Clustering ontology-based metadata in the semantic web, in *PKDD '02: Proc. of the 6th European Conf. on Principles of Data Mining and Knowledge Discovery* (Springer-Verlag, London, UK, 2002), pp. 348–360.
25. D. McDermott, Estimated-regression planning for interactions with web services, in *Proc. of the 6th Int. Conf. on Artificial Intelligence Planning Systems* (AAAI Press, 2002), pp. 204–211.
26. S. McIlraith and T. C. Son, Adapting GOLOG for composition of semantic web services, in *Proc. of the 8th Int. Conf. on Knowledge Representation and Reasoning* (2002), pp. 482–493.
27. M. Nagarajan, K. Verma, A. P. Sheth, J. A. Miller and J. Lathem, Semantic interoperability of web services — Challenges and experiences, *IEEE Int. Conf. on Web Services* (*ICWS 2006*).
28. G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross and K. Miller, Wordnet: An on-line lexical database, *International Journal of Lexicography* **3** (1990) 235–244.
29. C. Okutan, and N. K. Cicekli, A monolithic approach to automated composition of semantic web services with the event calculus, *Knowledge-Based Systems* **23**(5) (2010) 440–454.
30. OWL-S 1.1, 2004, http://www.daml.org/services/owl-s/1.1/ [retrieved 27/10/12].
31. OWLS-TC, http://projects.semwebcentral.org/projects/owls-tc/ [retrieved 21/01/14].
32. M. Paolucci, T. Kawamura, T. R. Payne and K. P. Sycara, Semantic matching of web services capabilities, in *International Semantic Web Conference*, I. Horrocks, J. A. Hendler, I. Horrocks and J. A. Hendler (eds.), Vol. 2342 of Lecture Notes in Computer Science (Springer, 2002), pp. 333–347.
33. M. Paolucci, A. Ankolekar, N. Srinivasan and K. Sycara, The DAML-S virtual machine, in *The SemanticWeb* (*ISWC 2003*), Vol. 2870 of Lecture Notes in Computer Science (Springer Berlin/Heidelberg, 2003), pp. 290–305.
34. M. Pistore, A. Marconi, P. Bertoli and P. Traverso, Automated composition of web services by planning at the knowledge level, in *Proc. of 19th Int. Joint Conferences on Artificial Intelligence* (2005), pp. 1252–1259.

35. S. R. Ponnekanti and A. Fox, SWORD: A developer toolkit for web service composition, in *Proc. of the 11th Int. WWW Conf.* (*WWW2002*) (Elsevier, Honolulu, HI, USA, 2002), pp. 83–107.
36. Radiant, WSDL-S/SAWSDL Annotation Tool, http://lsdis.cs.uga.edu/projects/meteors/downloads/index.php?page=1 [retrieved 20/01/13].
37. J. Rao and X. Su, A survey of automated web service composition methods, in *Proc. of the 1st Int. Workshop on Semantic Web Services and Web Process Composition* (*SWSWPC 2004*) (2004), pp. 43–54.
38. G. Salton, C. Buckley and C. T. Yu, An evaluation of term dependence models in information retrieval, in *Research and Development in Information Retrieval* (*LNCS 146*) (1982).
39. SAWSDL, Semantic Annotations for WSDL, 2007, http://www.w3.org/2002/ws/sawsdl/ [retrieved 27/10/10].
40. SAWSDL4J, http://sawsdl4j.sourceforge.net/ [retrieved 20/01/13].
41. E. Sirin, B. Parsia, D. Wu, J. Hendler and D. Nau, HTN planning for web service composition using SHOP2, *Journal of Web Semantics* **1**(4) (2004) 377–396.
42. SOAP, Simple Object Access Protocol, 2007, http://www.w3.org/TR/soap/ [retrieved 12/05/11].
43. G. Tsatsaronis, I. Varlamis and M. Vazirgiannis, Text relatedness based on a word thesaurus, *J. Artif. Intell. Res.* (*JAIR*) **37** (2010) 1–39.
44. L. Wang and X. Liu, A new model of evaluating concept similarity, *Knowledge-Based Systems* **21**(8) (2008) 842–846.
45. WebSphere business integration server foundation, http://www01.ibm.com/software/integration/wbisf/ [retrieved 12/05/11].
46. WSDL, Web Service Description Language, 2001, http://www.w3.org/TR/wsdl [retrieved 12/05/13].