PORSCE II: Using Planning for Semantic Web Service Composition

Ourania Hatzi[1] , Georgios Meditskos[2], Dimitris Vrakas[2], Nick Bassiliades[2], Dimosthenis Anagnostopoulos[1], Ioannis Vlahavas[2]

[1]Department of Informatics and Telematics, Harokopio University of Athens, Greece
[2]Department of Informatics, Aristotle University of Thessaloniki, Greece
{raniah, dimosthe}@hua.gr, {gmeditsk, dvrakas, nbassili, vlahavas}@csd.auth.gr

Abstract

This paper presents PORSCE II, an integrated system that performs automatic semantic web service composition through planning. In order to achieve that, an essential step is the translation of the web service composition problem into a planning problem. The planning problem is then solved using external domain-independent planning systems, and the solutions are visualized and evaluated. The system exploits semantic information to enhance the translation and planning processes.

1. Introduction

Web services nowadays are essential parts of the World Wide Web, as they accommodate interoperability between heterogeneous systems. However, in many cases, the need for complex and integrated service functionality cannot be fulfilled by a simple atomic web service, leading to the requirement for web service composition. The task of web service composition becomes significantly difficult, time-consuming and inefficient as the number of available atomic services increases continuously. Therefore, the possibility to automate the web service composition process is proved essential.

Automated web service composition is significantly facilitated by the development of the Semantic Web, since the existence of semantic information permits composition using intelligent techniques, such as AI Planning. Semantic description of web services is accommodated through the development of a number of standards such as OWL-S [6], WSMO [11], SAWSDL [13] and WSDL-S [12].

PORSCE II aims at automated semantic web service composition through planning with semantic relaxation. The first and very significant step in this process involves translation of the web service composition problem to a planning problem. This translation takes place between the most prominent standards in each area: OWL-S [6] for semantic description of web services and PDDL [5] for definition of planning domains and problems. According to user preferences, the translation process may take into account semantics, resulting from the semantic analysis of the domain; if so, semantically equivalent or relevant concepts are also included, in order to cope with cases when no exact plans can be found. The result of the transformation process is a fully formulated planning problem which incorporates all the required semantic information. PORSCE II consequently exports the planning problem to PDDL and invokes external planning systems to acquire plans, which constitute descriptions of the desired complex service. Each plan is evaluated in terms of statistic and accuracy measures. Finally, the system integrates a visual component which accommodates plan visualization and modification.

The rest of the paper is organized as follows: Section 2 discusses some related work, Section 3 provides an overview of the OWL-S standard, while Section 4 outlines the system architecture. Section 5 elaborates on the translation process, including semantic analysis and relaxation performed in the system. Section 6 presents the rest of the system operations. Section 7 presents a case study and performance evaluation and finally, Section 8 concludes the paper and poses future directions.

2. Related Work

One of the first systems that attempted automatic web service composition is SHOP-2 [15]. The system uses services descriptions in DAML-S, the predecessor of OWL-S, and performs HTN planning to solve the problem. The disadvantage of this approach lies in the fact that the planning process, due to its hierarchical nature, requires given decomposition rules, or methods, as they are referred to, which have to be encoded in advance with the help of a DAML-S process ontology.

OWLS-Xplan [16] 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 the complex services. The system is PDDL compliant, as the authors have developed an XML dialect of PDDL called PDDXML. However, semantic information provided from domain ontologies is not utilized; therefore the planning module requires exact matching for service inputs and outputs.

Other approaches for automatic web service composition are not further discussed here either because they do not deal with the important issue of translating semantic web service descriptions into planning terms or because they require some prior, domain-specific knowledge of the composition issues.

The main advantage of the proposed framework with respect to the aforementioned systems is the extended utilization of semantic information, in order to perform planning under semantic relaxation and find approximate solutions. Furthermore, PORSCE II does not require any prior, domain-specific knowledge to form valid, desired complex services, apart from the OWL-S descriptions of the atomic web services and the corresponding ontologies. Finally, the system is able to handle cases of service failure through simple service replacement, without the obligation to perform planning again.

3. OWL-S

OWL-S is an upper ontology based on OWL [14], created in the context of the Semantic Web in order to describe knowledge concerning semantic web services. It is used in combination with additional ontologies, which organize the concepts appearing in the OWL-S descriptions. The use of OWL-S renders the semantics of the descriptions machine comprehensible; therefore it enables intelligent agents to discover, invoke and compose web services automatically.

A web service description in OWL-S is comprised of three parts [6]:

- *Service Profile:* describes what the service accomplishes, limitations on service applicability and quality, and requirements that the service requester must satisfy to use the service.
- *Process Model:* describes the way a client can communicate and use the service.
- *Service Grounding:* specifies the details of how an agent can access a service, such as a communication protocols and message formats.

The approach presented here utilizes the semantic information contained in the *Service Profile* of a specific web service, along with the corresponding ontologies, in order to translate the description in planning terms. An example of an OWL-S Profile and the correspondences between its elements of interest and the planning terms they are translated into is provided in Section 5.1.

4. System Overview and Architecture

PORSCE II is the evolution of the prototype system PORSCE [7]. The core translation component exists in both systems; however, PORSCE II aims at a higher degree of integration as it additionally contains a visual interface, more elaborate relevance metrics, complex service accuracy assessment and the ability to modify complex web services. Furthermore, PORSCE II adopts a different way of modeling the web service composition as a planning problem, which reduces the complexity of the planning problem, thus accelerating the planning process.

In order to highlight the planner independency of PORSCE II, which enables the use of any domain independent planning system based on PDDL, another external planner has been included in addition to the original one.

The main features of the integrated system are:

- Translation of OWL-S atomic web service descriptions into planning operators.
- Interaction with the user in order to acquire their preferences regarding the complex service and desired metrics for concept relevance.
- Enhancing the planning domain and problem with semantically similar concepts.
- Exporting the web service composition problem as a planning domain and problem in PDDL.
- Providing solutions by invoking external planners.
- Assessing the accuracy of the complex services.
- Visualizing and modifying the solution.

PORSCE II comprises of the OWL-S Parser, the Transformation Component, the OWL Ontology Manager (OOM), the Visualizer and the Service Replacement Component. An overview of the architecture and the interactions among the components is depicted in Fig. 1.

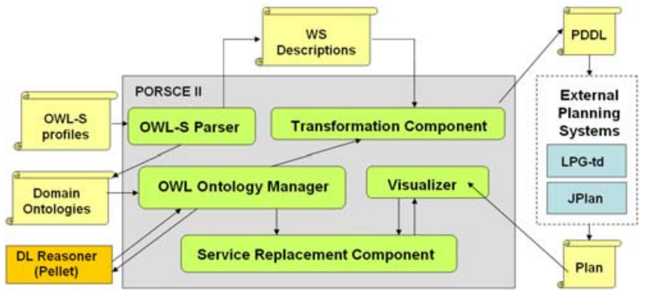


Fig. 1 The architecture of PORSCE II.

The OWL-S Parser is responsible for parsing a set of OWL-S web service profiles and determining the corresponding ontologies that the concepts appearing in the web service descriptions belong to. The OWL Ontology Manager (OOM), utilizing the Pellet DL Reasoner [2], applies the selected algorithm for discovering concepts that are similar to a query concept. The Transformation Component is responsible for a number of operations that result in the formulation of the planning problem from the initial web service composition problem, and its consequent solving. The purpose of the Visualizer is to provide the user with a visual representation of the plan, which in fact is the description of the complex service. Finally, the Service Replacement Component enables the user to replace a specific atomic web service in the complex service sequence. Details on the system functionality regarding the translation process and the rest of its operations are provided in Sections 5 and 6.

PORSCE II is implemented in Java and it is available online, along with example problems, at http://www.dit.hua.gr/~raniah/porsceII_en.html

5. Transformation Process

The transformation process includes the translation of the web service composition problem into a planning problem and its possible enhancement with semantic information. The process starts at the OWL-S Parser, which parses the OWL-S profiles of the available atomic web services and forwards them to the Transformation Component. The Transformation Component is responsible for a number of operations, including translating the web service descriptions received from the OWL-S Parser to planning operators and enhancing them with similar concepts derived from the OOM. Moreover, it interacts with the user in order to formulate the planning problem, and exports both the planning domain and problem to PDDL.

5.1. OWL-S to PDDL Translation

A planning problem in PORSCE II, in accordance with the STRIPS notation [4], is a tuple <I,A,G> where I is the initial state, A is the set of available actions that can be used to modify states, and G is the set of goals. 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 after the application of A_i, denoted as prec(A_i), add(A_i) and del(A_i) respectively

A straightforward solution adopted by PORSCE II for mapping the web service composition problem to a planning problem is the following: Let *IC* be the set of concepts that the user wishes to provide to the complex service and *GC* its desired outputs (goals). If *O* denotes the set of all the available concepts in the ontology, then $IC \subseteq O$, $GC \subseteq O$ and $IC \cap GC \equiv \varnothing$. The inputs that the user wishes to provide formulate the initial state, while the desired outputs of the complex service formulate the goals of the problem: I = *IC* and G=*GC*. Both the input and output sets are provided externally by the user.

The available OWL-S web service profiles are used in order to obtain the planning operators: each web service description WSDi is translated to an operator Ai, using the information in the each profile (Fig. 2).

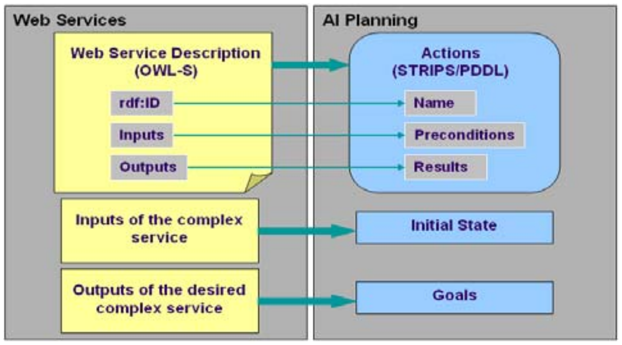


Fig. 2 Mapping web services to planning.

More specifically:

- The name of the action is the rdf:ID of the profile:
$$\text{name}(\text{A}_i) = WSD_i.\text{ID}$$
- The preconditions of the action are formed based on the service's input definitions (concepts):
$$\text{prec}(\text{A}_i) \equiv \bigcup_{k=1}^{n} \{\text{WSD}_i.\text{hasInput}_k\}$$
- The add effects of the action comprises of the service's output definitions (concepts):
$$\text{add}(\text{A}_i) \equiv \bigcup_{k=1}^{m} \{\text{WSD}_i.\text{hasOutput}_k\}$$
- The delete list is left empty, since in the current study only services that do not have any negative effects in the world model are dealt with: $\text{del}(\text{A}_i) \equiv \varnothing$

An example of an OWL-S to PDDL transformation is presented in Fig. 3, where the mapping presented above is marked. The web service description at hand concerns a web service that accepts as input the activity Sightseeing and presents the user with areas that offer this activity.

5.2. Semantic Analysis

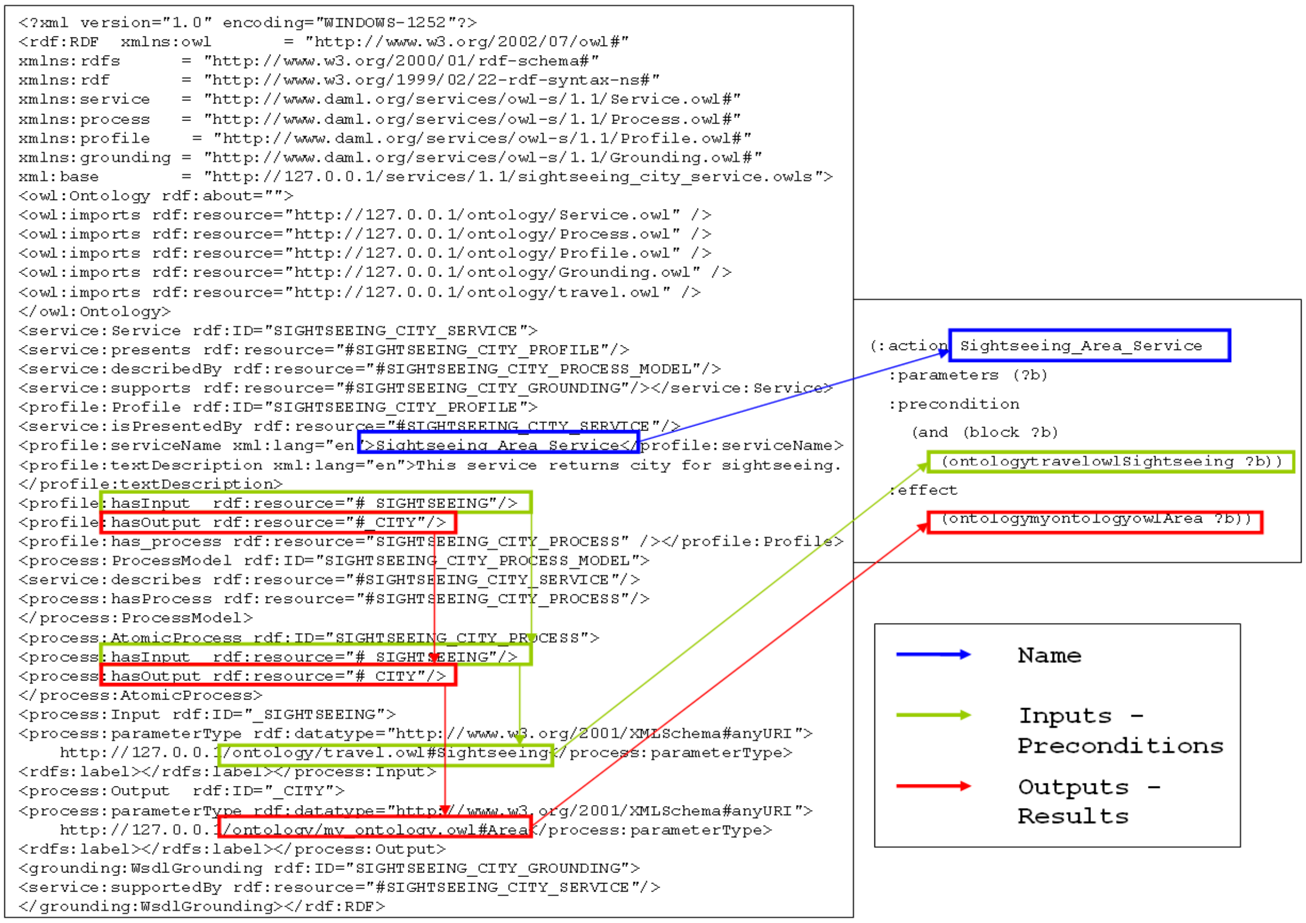
The step of semantic analysis, parallel to the transformation process, enables the system to translate the web service composition problem to a planning problem by taking into account semantic information. This step is implemented by the OWL Ontology Manager (OOM). During translation, the OOM is used extensively for performing semantic relaxation, which is useful in cases when an exact input/output matching plan is not available. The OOM locates equivalent and semantically relevant concepts; therefore, approximate plans can be created.

In our approach, two ontology concepts are considered semantically similar if and only if

- ❖ they have a *hierarchical relationship*
- ❖ their *semantic distance* does not exceed a threshold

As far as the *hierarchical relationship* is concerned, four hierarchical filters are used for its definition for two ontology concepts *A* and *B*:

- *exact*(*A*, *B*): The two concepts should have the same URI or they should be equivalent, in terms of OWL class equivalence, i.e. $A = B \vee A \equiv B$.
- *plugin*(*A*, *B*): The concept *A* should be subsumed by the concept *B*, i.e. $A \sqsubseteq B$.
- *subsume*(*A*, *B*): The concept *A* should subsume the concept *B*, i.e. $B \sqsubseteq A$. In both the *plugin* and the *subsume* filters the subsumption relationships of equivalent concepts are not considered.
- *sibling*(*A*, *B*): The two concepts should neither have a hierarchical relationship, nor be disjoint; instead, they should have a common superclass *T*, such as $A \sqsubseteq T \wedge B \sqsubseteq T$.



```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF  xmlns:owl       = "http://www.w3.org/2002/07/owl#"
xmlns:rdfs      = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf       = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:service   = "http://www.daml.org/services/owl-s/1.1/Service.owl#"
xmlns:process   = "http://www.daml.org/services/owl-s/1.1/Process.owl#"
xmlns:profile    = "http://www.daml.org/services/owl-s/1.1/Profile.owl#"
xmlns:grounding = "http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
xml:base        = "http://127.0.0.1/services/1.1/sightseeing_city_service.owls">
<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://127.0.0.1/ontology/Service.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Process.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Profile.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Grounding.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/travel.owl" />
</owl:Ontology>
<service:Service rdf:ID="SIGHTSEEING_CITY_SERVICE">
<service:presents rdf:resource="#SIGHTSEEING_CITY_PROFILE"/>
<service:describedBy rdf:resource="#SIGHTSEEING_CITY_PROCESS_MODEL"/>
<service:supports rdf:resource="#SIGHTSEEING_CITY_GROUNDING"/></service:Service>
<profile:Profile rdf:ID="SIGHTSEEING_CITY_PROFILE">
<service:isPresentedBy rdf:resource="#SIGHTSEEING_CITY_SERVICE"/>
<profile:serviceName xml:lang="en">Sightseeing Area Service</profile:serviceName>
<profile:textDescription xml:lang="en">This service returns city for sightseeing.
</profile:textDescription>
<profile:hasInput  rdf:resource="# SIGHTSEEING"/>
<profile:hasOutput rdf:resource="#_CITY"/>
<profile:has_process rdf:resource="SIGHTSEEING_CITY_PROCESS" /></profile:Profile>
<process:ProcessModel rdf:ID="SIGHTSEEING_CITY_PROCESS_MODEL">
<service:describes rdf:resource="#SIGHTSEEING_CITY_SERVICE"/>
<process:hasProcess rdf:resource="#SIGHTSEEING_CITY_PROCESS"/>
</process:ProcessModel>
<process:AtomicProcess rdf:ID="SIGHTSEEING_CITY_PROCESS">
<process:hasInput  rdf:resource="# SIGHTSEEING"/>
<process:hasOutput rdf:resource="# CITY"/>
</process:AtomicProcess>
<process:Input rdf:ID="_SIGHTSEEING">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
    http://127.0.0.1/ontology/travel.owl#Sightseeing</process:parameterType>
<rdfs:label></rdfs:label></process:Input>
<process:Output  rdf:ID="_CITY">
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
    http://127.0.0.1/ontology/my_ontology.owl#Area</process:parameterType>
<rdfs:label></rdfs:label></process:Output>
<grounding:WsdlGrounding rdf:ID="SIGHTSEEING_CITY_GROUNDING">
<service:supportedBy rdf:resource="#SIGHTSEEING_CITY_SERVICE"/>
</grounding:WsdlGrounding></rdf:RDF>
```

```
(:action Sightseeing_Area_Service
  :parameters (?b)
  :precondition
    (and (block ?b)
      (ontologytravelowlSightseeing ?b))
  :effect
      (ontologymyontologyowlArea ?b))
```

Fig. 3 Example of an OWL-S to PDDL

The *semantic distance* between two ontology concepts can be calculated in PORSCE II using two methods:

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

The implementation of the *ec* distance between two concepts, denoted as $d_{ec}(A, B)$, returns a value between 0 and 1, with 1 denoting absolute mismatch, and considers the following cases:

- $exact(A, B) \Rightarrow d_{ec}(A, B) = 0$: equivalent concepts
- $A \sqcap B \sqsubseteq \perp \Rightarrow d_{ec}(A, B) = 1$: disjoint concepts
- $plugin(A, B) \vee subsume(A, B) \Rightarrow d_{ec}(A, B) = p/p_{\max}$. If there is a hierarchical relationship between the two concepts, the distance is equal to the number of edges in their shortest path (p) normalized to [0..1] using the maximum *ec* distance (p_{max}) found in the ontology, which can be approximated as $p_{max} = 2h - 1$, where h is the maximum edge distance from a leaf concept to the owl:Thing concept ($\top$).
- $sibling(A, B) \Rightarrow d_{ec}(A, B) = \min[d_{ec}(A, T) + d_{ec}(B, T)]$: the concepts have a sibling relationship and the *ec* distance is the minimum of the sum of the *ec* distances of each concept from the least common ancestor T. Note that the owl:Thing is not considered as a common ancestor, since $\forall A, B : A \sqsubseteq \top \wedge B \sqsubseteq \top$ and therefore, no special structural knowledge is provided.

The Upwards Cotopic Distance, denoted as $d_{uc}(A, B)$, is defined in terms of the upwards cotopic measure, denoted as $uc(A)$ that represents the set of the superclasses of the concept A, including A itself [10]. In PORSCE II, the upwards cotopic distance definition has been modified in order to incorporate the semantics of an ontology hierarchy. More specifically, the owl:Thing concept is not considered in the *uc* measure, while the union and intersection set operators take into account the concept equivalence semantics. In that way concept set multiplicity is ignored, that is, if $A \equiv B$, then $\{A, B, C\} = \{A, C\} \vee \{B, C\}$, and the concept set membership is semantically checked, that is, if $A \equiv B$ and $D = \{A\}$, then $A \in D \wedge B \in D$. Based on these remarks, the upwards cotopic distance is defined as

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

If two concepts have only the owl:Thing class as the common superclass or they are disjoint, then their distance equals to 1; otherwise, if the two concepts have a hierarchical relationship, then $d_{uc}(A, B) \in [0..1)$.

5.3. Semantic Awareness and Relaxation

The representation of the web service composition as a planning problem is empowered if the planning system is aware of semantic similarities among syntactically different concepts (semantic awareness). The solution adopted by PORSCE II involves enhancing the domain and problem description with all the required semantic information in a pre-processing phase and letting the planner handle it as a classical planning problem. PORSCE II adopts this solution in order to: a) be able to use any planner, compliant with PDDL, as the semantic enhancement applied to the domain remains transparent to the planner, and b) minimize the interactions between the planner and the OOM, which introduce an overhead on the planning time.

In the pre-processing phase, the system uses the OOM in order to acquire all the semantically relevant concepts for both the facts of the initial state and the outputs of the operators, discovered by the semantic analysis process described in the previous section. The enhancement of the problem by PORSCE II is based on the following rules:

- The original concepts of the initial state together with the semantically equivalent and similar concepts form a new set of facts noted as the Expanded Initial State (EIS).
- The goals of the problem remain the same.
- The Enhanced Operator Set (EOS) is produced, by altering the description of each operator, while preserving the initial size of the set. More specifically, the effects list of each operator is enhanced by including all the equivalent and semantically similar concepts for the concepts in the initial effects list.

Suppose, for example, that the initial state I of the problem is the following:

```
I = {Sightseeing, Dates}
```

and that there are only the following two operators:

```
CityHotelMapService:
  prec={City, Hotel}, effect={Map}
SightSeeingAreaService:
  prec={Sightseeing}, effect={Area}
```

The OOM for a given distance metric and threshold discovers the following relevant concepts:

```
Dates ≈ Duration
Area ≈ County,
Map ≈ GPSRoute
```

The pre-processor alters the problem definition to the following:

```
EIS: {Sightseeing, Dates, Duration}
EOS: CityHotelMapService:
          prec={City, Hotel}
          effect={Map, GPSRoute}
     SightSeeingAreaService:
          prec={Sightseeing}
          effect={Area, County}
```

The new problem, namely <EIS,EOS,G> is encoded into PDDL and forwarded to the planning system in order to acquire a solution. Note that the semantic information is encoded in such a way that it is transparent to the external planning systems, which can solve the problem as any other classical planning problem.

6. Solution and Integration

PORSCE II aims at integrating the composition process, including solving the problem through invocation of external planning systems, visualization, evaluation and modification of the solutions.

6.1. Acquiring Solutions

Since the transformation process results in the export of both the planning domain and problem in PDDL, any PDDL-compliant domain independent external planning system can be used.

Currently, two different planning modules have been incorporated in the system: JPlan [1], which is an open-source Java implementation of Graphplan and LPG-*td* [8]. Both planners proved to be remarkably fast and can handle a respectable number of operators, which is very important as the number of available web services is expected to increase significantly over time. After the planning process is completed, JPlan provides the plan, in its own format, which comprises of a simple sequential list of actions. LPG-*td*, on the other hand, provides the plan in a format that complies with PDDL+. The plan in this case might not be sequential, but structured in levels; actions belonging to the same level can be executed in an arbitrary sequence, however all actions of a certain level must be completed before any action of the following level can be executed. Subsequently, these plans are visualized and their accuracy is evaluated.

6.2. Complex Service Accuracy Assessment

Semantic relaxation and the use of multiple planners may produce a number of complex services, for which statistics and quality metrics have to be calculated. Such metrics include the number of actions and the number of levels in the plan, as well as a plan distance quality metric, which indicates the accuracy of the plan, when semantic relaxation takes place.

For the calculation of the plan semantic distance, each concept appearing in the inputs or outputs of the actions of the plan is annotated by the OOM with a semantic distance d_i with respect to the original concept it was derived from, using the selected similarity metric. A concept distance of 0 reveals identical or equivalent concepts. Additionally, each concept is annotated with a weight w_i, which represents the kind of hierarchical relationship to the original concept, as in some cases certain hierarchical relationships might be more desirable than others.

These values are combined to form a plan semantic distance. For example, when the upwards cotopic distance metric is used, the plan semantic distance is calculated as a weighted product of these concepts, as the product represents better the semantic distance in this case:

$$PSD_{uc} = \prod_{i=0}^{n} w_i d_i, d_i \neq 0$$

The plan accuracy metric in both cases is calculated as 1-*PSD*; therefore, if there is exact input to output matching, or if only equivalent concepts are used, then the plan quality metric value is 1, while it decreases as the plan becomes less accurate.

6.3. Visualization and Modification

The Visualizer enhances comprehensibility by providing a visual representation of the complex service and facilitates plan manipulation. The complex service is represented as a schema of simple service invocations, showing inputs and outputs (Fig. 7, 8).
The Visualizer module invokes and interacts closely with another module of the system, the Service Replacement Component, which discovers all actions that could be used alternatively instead of the chosen one, using advice from the OOM for equivalent and relative concepts. An action A is considered an alternative for an action Q of the plan as far as it does not disturb the plan sequence and the intermediate states, that is prec(A) $\subseteq$ prec(Q) and add(A) $\supseteq$ add(Q). The selected alternative service substitutes the original one both in the plan and in the visualization.

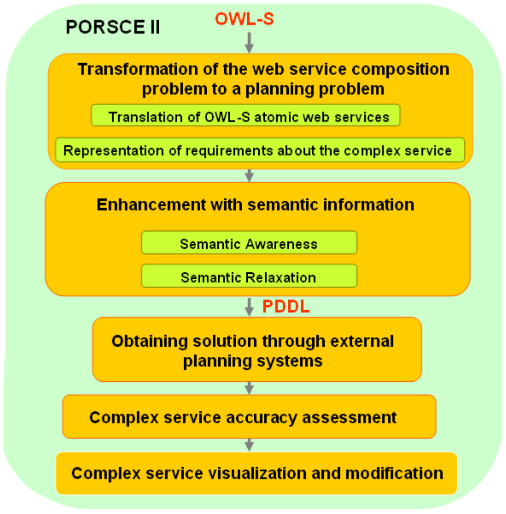

PORSCE II
OWL-S
Transformation of the web service composition problem to a planning problem
Translation of OWL-S atomic web services
Representation of requirements about the complex service
Enhancement with semantic information
Semantic Awareness
Semantic Relaxation
PDDL
Obtaining solution through external planning systems
Complex service accuracy assessment
Complex service visualization and modification

Fig. 4 The steps for the demonstration.

7. Demonstration and System Evaluation

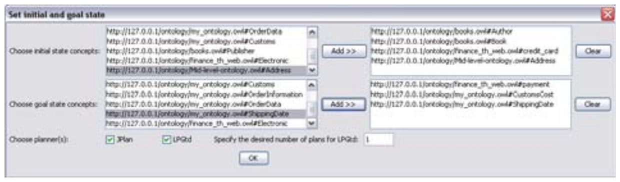
This section aims at demonstrating the use and evaluating the performance of the system through a case study, following the general course depicted in Fig. 4.

The test sets used to perform experiments were obtained from the OWLS-TC version 2.2 revision 1 [3], while several service descriptions were modified or added to these domains, accommodating the demonstration of the capabilities of the system. The web services that were modified or added to the domain are depicted in Table 1.

Table 1. The added / modified web services.

Service	Inputs	Outputs
`BookToPublisher`	`Book, Author`	`Publisher`
`CreditCardCharge`	`OrderData, CreditCard`	`Payment`
`ElectronicOrder`	`Electronic`	`OrderData`
`PublisherElectronicOrder`	`PublisherInfo`	`OrderData`
`ElectronicOrderInfo`	`Electronic`	`OrderInformation`
`Shipping`	`Address, OrderData`	`ShippingDate`
`WaysOfOrder`	`Publisher`	`Electronic`
`CustomsCost`	`Publisher, OrderData`	`CustomsCost`

The transformation of the web service composition problem to a planning problem includes translating all available OWL-S atomic web services, including the aforementioned ones, to PDDL operators. It also incorporates the representation of the requirements about the complex service, which the user can express through a dialog interface such as the one depicted in Fig. 5.


Set initial and goal state

Choose initial state concepts:
http://127.0.0.1/ontology/my_ontology.owl#OrderData
http://127.0.0.1/ontology/my_ontology.owl#Customs
http://127.0.0.1/ontology/books.owl#Publisher
http://127.0.0.1/ontology/finance_th_web.owl#Electronic
http://127.0.0.1/ontology/Mid-level-ontology.owl#Address

Add >>

http://127.0.0.1/ontology/books.owl#Author
http://127.0.0.1/ontology/books.owl#Book
http://127.0.0.1/ontology/finance_th_web.owl#credit_card
http://127.0.0.1/ontology/Mid-level-ontology.owl#Address

Clear

Choose goal state concepts:
http://127.0.0.1/ontology/my_ontology.owl#Customs
http://127.0.0.1/ontology/my_ontology.owl#OrderInformation
http://127.0.0.1/ontology/my_ontology.owl#OrderData
http://127.0.0.1/ontology/my_ontology.owl#ShippingDate
http://127.0.0.1/ontology/finance_th_web.owl#Electronic

Add >>

http://127.0.0.1/ontology/finance_th_web.owl#payment
http://127.0.0.1/ontology/my_ontology.owl#CustomsCost
http://127.0.0.1/ontology/my_ontology.owl#ShippingDate

Clear

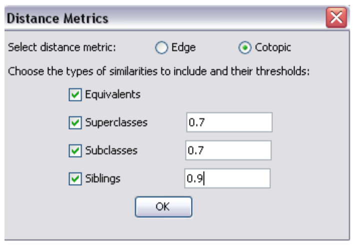
Choose planner(s): ✓ JPlan ✓ LPG-td Specify the desired number of plans for LPG-td: 1

OK

Fig. 5. Defining initial and goal states and desired planners.

The scenario implemented here concerns the electronic purchase of a book. The user provides as inputs a book title and author, credit card info and the address that the book will be shipped to, and requires a charge to credit card for the purchase, as well as the shipping dates and the customs cost for the specific item. The initial state corresponds to the inputs of the complex service, while the goal state represents the desired complex service outcome.

The next step is optional semantic relaxation, performed through semantic enhancement. The user defines semantic metrics and thresholds through the interface depicted in Fig. 6.


Distance Metrics

Select distance metric: Edge [✓ Cotopic]

Choose the types of similarities to include and their thresholds:

☑ Equivalents	
☑ Superclasses	0.7
☑ Subclasses	0.7
☑ Siblings	0.9

OK

Fig. 6. The semantic enhancement interface.

At this point, the system exports the formulated (and possibly semantically enhanced) planning domain and problem to PDDL. Consequently, it invokes external planners to acquire solutions.

The plans produced by JPlan and LPG-td for the specific case study using the operator set described above, without including any semantically relevant concepts are presented in Fig. 7.

While exact matching of input to output concepts is obligatory in the classical planning domains, in the web services world the case can be different, as it is preferable to present the user with a complex service that approximates the required functionality than to present no service at all. The semantically similar concepts obtained from the OWL Ontology Manager enable the system to compose alternative services that approximate the desired one in case there are no exact matches, by performing semantically relaxed concept matching. Such an approximate service for the specific case study is presented in Fig. 8. The calculated accuracy of this service is different from the accurate ones presented in Fig. 7.

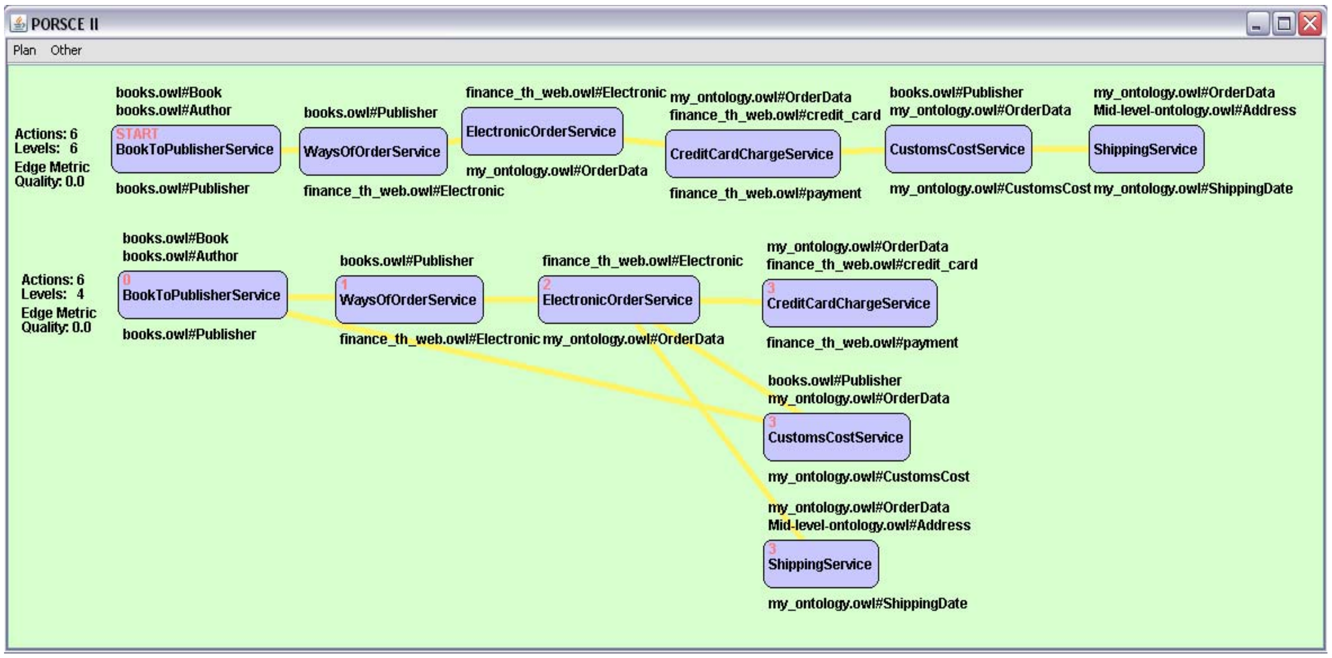

PORSCE II
Plan Other
Actions: 6
Levels: 6
Edge Metric Quality: 0.0
books.owl#Book
books.owl#Author
START
BookToPublisherService
books.owl#Publisher
books.owl#Publisher
WaysOfOrderService
finance_th_web.owl#Electronic
finance_th_web.owl#Electronic
ElectronicOrderService
my_ontology.owl#OrderData
my_ontology.owl#OrderData
finance_th_web.owl#credit_card
CreditCardChargeService
finance_th_web.owl#payment
books.owl#Publisher
my_ontology.owl#OrderData
CustomsCostService
my_ontology.owl#CustomsCost
my_ontology.owl#OrderData
Mid-level-ontology.owl#Address
ShippingService
my_ontology.owl#ShippingDate
Actions: 6
Levels: 4
Edge Metric Quality: 0.0
books.owl#Book
books.owl#Author
0
BookToPublisherService
books.owl#Publisher
books.owl#Publisher
1
WaysOfOrderService
finance_th_web.owl#Electronic
finance_th_web.owl#Electronic
2
ElectronicOrderService
my_ontology.owl#OrderData
my_ontology.owl#OrderData
finance_th_web.owl#credit_card
3
CreditCardChargeService
finance_th_web.owl#payment
books.owl#Publisher
my_ontology.owl#OrderData
3
CustomsCostService
my_ontology.owl#CustomsCost
my_ontology.owl#OrderData
Mid-level-ontology.owl#Address
3
ShippingService
my_ontology.owl#ShippingDate

Fig. 7. The plans from JPlan (top) and LPG-td (bottom) for the specific case study.

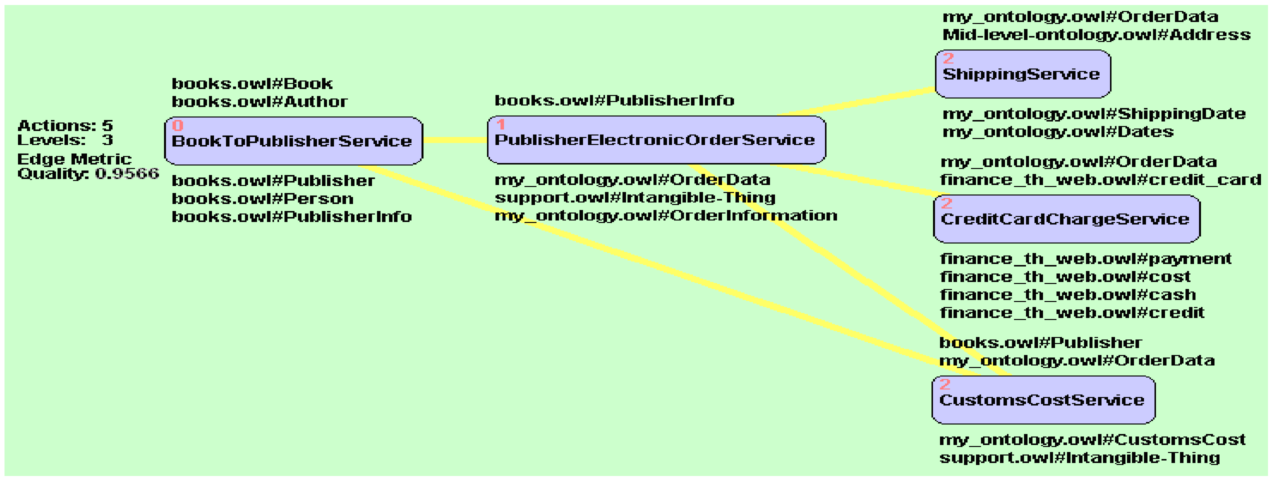

Actions: 5
Levels: 3
Edge Metric Quality: 0.9566
books.owl#Book
books.owl#Author
0
BookToPublisherService
books.owl#Publisher
books.owl#Person
books.owl#PublisherInfo
books.owl#PublisherInfo
1
PublisherElectronicOrderService
my_ontology.owl#OrderData
support.owl#Intangible-Thing
my_ontology.owl#OrderInformation
my_ontology.owl#OrderData
Mid-level-ontology.owl#Address
2
ShippingService
my_ontology.owl#ShippingDate
my_ontology.owl#Dates
my_ontology.owl#OrderData
finance_th_web.owl#credit_card
2
CreditCardChargeService
finance_th_web.owl#payment
finance_th_web.owl#cost
finance_th_web.owl#cash
finance_th_web.owl#credit
books.owl#Publisher
my_ontology.owl#OrderData
2
CustomsCostService
my_ontology.owl#CustomsCost
support.owl#Intangible-Thing

Fig. 8. Approximate complex service.

In order to study the behavior of the system as the number of available web services increases, web service profiles were added to the domain progressively in batches. The time performance results presented in Table 2 were obtained from a number of runs of the system on a machine with Dual-Core AMD Opteron Processor at 2.20GHz with 1GB of RAM memory and concern times for preprocessing, transformation of the OWL-S service profiles to PDDL actions and planning using LPG-td.

Measurements took place for domains of different sizes, namely 10, 100, 500 and 1000 OWL-S profiles. Some of the experiments were performed without semantic relaxation (X), while others were performed with semantic relaxation using either the edge-counting distance metric (E) or the upwards cotopic metric (C). The preprocessing time did not show significant fluctuation, as it depends only on the number and structure of the processed ontologies and not on the number of available web services. The total transformation time evidently increased as the number of available web services increased, however the average transformation time per web service profile converged to approximately 0.8 seconds for the exact matching and the edge-counting distance metric cases. In the upwards cotopic metric distance, the increase in the average transformation time is significant as available web services increase, due to the higher complexity of the algorithm used for the calculation of the upwards cotopic relevance between two concepts. As far as average planning time is concerned, LPG-td shows an increase in planning time as the number of actions increases, however it is still proved remarkably fast.

Table 2. Time measurements in milliseconds.

umber of web rvices		10	100	500	1000
Preprocessing time		5857	6104	5875	5703
Total transformation time	X	4594	70062	350836	792109
	E	4531	75725	335477	796797
	C	4585	74688	728633	3901141
Transformation time per web service	X	459	700	702	792
	E	453	671	757	797
	C	459	746	1457	3901
Planning time (LPGtd)	X	1	13	16	17
	E	4	6	15	16
	C	3	5	16	16

8. Conclusions and Future Work

This paper presented PORSCE II, which combines planning with semantic object relevance in order to approach the semantic web service composition problem. Each web service composition problem is translated into a planning problem, possibly enhanced with semantically relevant concepts and exported to PDDL. The system integrates external planning system which perform planning with the desired degree of semantic relaxation. The obtained plans, which represent descriptions of the desired complex web service, are evaluated, visualized and modified.

Future goals include the extension of the system in order to translate the plan describing the complex service into OWL-S so the complex web service can be invoked and provide feedback. In addition, another goal concerns incorporating the quality distance metric with plan statistics in a common metric. Furthermore, integration with the VLEPPO system [9] is a promising future direction, in order to accommodate design and solving of the web service composition problems. Finally, it lies in our immediate plans to study ways to enhance the services representation and explore the ability to produce various complex services according to non-functional properties.

References

[1] JPlan: Java Graphplan Implementation, http://sourceforge.net/projects/jplan

[2] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz, Pellet: A Practical OWL DL Reasoner, J. Web Semantics, 2007

[3] OWLS-TC version 2.2 revision 1, http://projects.semwebcentral.org/ projects/owls-tc/

[4] R. Fikes, N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving", Artificial Intelligence, Vol 2 (1971), 189-208.

[5] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, "PDDL -- the Planning Domain Definition Language". Technical report, Yale University, New Haven, CT (1998).

[6] OWL-S 1.1. http://www.daml.org/services/owl-s/1.1/

[7] O. Hatzi, G. Meditskos, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, I. Vlahavas, A Synergy of Planning and Ontology Concept Ranking for Semantic Web Service Composition, IBERAMIA 2008, 11th Ibero-American Conference on AI, Lisbon, Portugal, October 14-17, 2008. Proceedings. Lecture Notes in Computer Science 5290 Springer 2008, pp 42-51.

[8] A. Gerevini, A. Saetti, I. Serina, LPG-TD: a Fully Automated Planner for PDDL2.2 Domains" (short paper), in International Planning Competition, 14th Int. Conference on Automated Planning and Scheduling (ICAPS-04).

[9] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, I. Vlahavas, VLEPpO: A Visual Language for Problem Representation, 26th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2007), Roman Bartak (Ed.), pp. 60 – 66.

[10] A. Maedche and V. Zacharias, Clustering Ontology-Based Metadata in the Semantic Web, European Conf. Principles of Data Mining and Knowledge Discovery, 2002.

[11] WSMO, http://www.wsmo.org/

[12] WSDL-S, http://www.w3.org/Submission/WSDL-S/

[13] SAWSDL, http://www.w3.org/2002/ws/sawsdl/

[14] OWL, http://www.w3.org/TR/owl-ref/

[15] E. Sirin, B. Parsia, D. Wu, J. Hendler and D. Nau, 2004. HTN planning for web service composition using SHOP2. Journal of Web Semantics, 1(4) 377–396.

[16] M. Klusch, A. Gerber, M. Schmidt: Semantic Web Service Composition Planning with OWLS-XPlan. AAAI Fall Symposium on Semantic Web and Agents, USA, 2005.