An Integrated Approach to Automated Semantic Web Service Composition through Planning

Ourania Hatzi, Dimitris Vrakas, Mara Nikolaidou, Nick Bassiliades, Dimosthenis Anagnostopoulos, Ioannis Vlahavas

Abstract— The paper presents an integrated approach for automated semantic web service composition using AI planning techniques. An important advantage of this approach is that the composition process, as well as the discovery of the atomic services that take part in the composition, are significantly facilitated by the incorporation of semantic information. OWL-S web service descriptions are transformed into a planning problem described in a standardized fashion using PDDL, while semantic information is used for the enhancement of the composition process as well as for approximating the optimal composite service when exact solutions are not found. Solving, visualization, manipulation and evaluation of the produced composite services are accomplished, while, unlike other systems, independence from specific planners is maintained. Implementation was performed through the development and integration of two software systems, namely PORSCE II and VLEPPO. PORSCE II is responsible for the transformation process, semantic enhancement and management of the results. VLEPPO is a general-purpose planning system used to automatically acquire solutions for the problem by invoking external planners. A case study is also presented to demonstrate the functionality, performance and potential of the approach.

Index Terms— Intelligent Web Services and Semantic Web, Services Composition, Composite Web Services

1 Introduction

WEB services provide a standardized way to achieve interoperability between heterogeneous software systems independently from underlying implementation technologies and platforms. However, the limited functionality offered by an atomic web service cannot always satisfy complex user needs and appropriately reflect intricate business processes [39]. Web service composition techniques attempt to solve such issues by combining and integrating suitable atomic web services into a composite one. The simplest option for web service composition is to perform it manually. In this case, a domain expert takes into account user requirements and browses through the available web services to eventually create a desired composite service. The composite service structure, as well as the atomic services taking part in the composition, are described statically. However, the most promising alternative is automated composition. The ability to perform web service discovery and composition automatically and dynamically is essential and has emerged as an important research topic [7][28][29][41].

Automated web service composition deals with the significant increase in the number of available services over time, as well as frequent changes in their definitions. It enables significantly faster responses to user queries for composite services, compared to the manual case. Also, it produces compositions up-to-date with the latest web service definitions, despite the dynamic environment.

Automated web service composition is commonly performed using AI techniques, especially AI planning [4][30][43][47][48]. Existing approaches [10][12][13] have successfully utilized intelligent techniques to dynamically locate services [49] and / or automatically compose them. They use WSDL or BPEL4WS descriptions emphasizing structural properties, which describe service interaction (e.g. service input – output).

The aforementioned approaches and the most common web service standards operate at the syntactic level. Incorporation of semantics can facilitate automation of both discovery and composition by eliminating syntactic barriers [27][38][42]. Semantics also allow the creation of approximate composite services and, consequently, the evaluation of their quality in terms of accuracy. Standards, such as OWL-S [31], provide the means to incorporate semantics in web service descriptions. Such semantic descriptions conform to ontologies which define relations among them. Enhanced composition features, such as approximation, are facilitated by the semantic information of OWL-S; however, existing tools and methodologies do not utilize this information [3][40].

Apart from syntactic and semantic knowledge, a third

- *O. Hatzi, M. Nikolaidou and D. Anagnostopoulos are with the Department of Informatics and Telematics, Harokopio University of Athens, Greece. E-mail: {raniah, mara, dimosthe}@hua.gr.*
- *D. Vrakas, N. Bassiliades and I. Vlahavas are with the Department of Informatics, Aristotle University of Thessaloniki, Greece. E-mail: dvrakas, nbassili, vlahavas}@csd.auth.gr.*

Manuscript received on June 24th, 2010; 1st revision on September 30th, 2010; 2nd revision on December 7th, 2010.

type called *contextual* or *pragmatic* knowledge is mentioned in the literature [45][46] as useful for providing *reasonable* compositions. Pragmatic knowledge does not concern the service itself, but describes the way the service relates to the satisfaction of the goals of the composition consumer [44]. For example, if several e-bookstore services offer a specific book, pragmatic knowledge could indicate that the cheapest offer should be accepted. The caption of such knowledge is not supported by current web service description standards, though the exploration of such issues is a very promising research direction.

We argue that automated composition of web services should exert the semantic information included in OWL-S to the full extent. Towards this direction, the paper presents an integrated approach for semantic web service composition, exploiting AI Planning techniques. It aims at providing enhanced composition functionality, including the expansion of the solution space with approximate composite services and the evaluation of solutions in terms of accuracy. The proposed approach is based on transforming the web service composition problem into a planning problem and solving it after enrichment with semantic information extracted from OWL-S. In this way, extensive research in AI planning can be applied to the area of web service composition. The produced domain is described using well-established standards, such as PDDL [19], while solutions may be acquired using a variety of external planners in a standard way. Independence from planning techniques and algorithms is provided, enabling us to take advantage of recent research advances. Solutions are transformed back to OWL-S descriptions, which are suitable for execution in any web service environment. The approach facilitates the composition process, even for non-expert users. The aforementioned activities are organized into a seven-step approach discussed in the paper.

The implementation of the approach is accommodated by the development and integration of two software systems. PORSCE II is responsible for all transformation procedures, semantic enhancement and management of the results. VLEPPO is a general-purpose, domain-independent system for the design and solving of planning problems. In the proposed framework, it is employed to automatically and flexibly acquire solutions to web service composition problems.

The rest of the paper is structured as follows: Section 2 discusses related work in the area of automated web service composition. Section 3 presents the proposed approach, identifying its discrete steps, the interaction between them and how they are implemented. Sections 4 and 5 elaborate on the PORSCE II and VLEPPO systems, respectively. Section 6 presents a case study and performance results and, finally, Section 7 provides conclusions and poses future directions.

2 Related Work

Web service composition approaches range from manual, where user intervention is required in every step of the process, to fully automated, where intervention is confined to defining user requirements for the desired composite service [3][4][5][6][7]. The criterion for this categorization is the degree of automation of two main aspects: structure of the composition and discovery of the atomic web services taking part in the composition. In manual approaches, composition is accomplished through standards, which describe complex business processes implemented through composite services; the most prominent is BPEL4WS [8]. Semi-automated approaches concern only parts of the process. For example, they automate only the selection of certain atomic web services that implement a manually defined composition scheme [9]. Finally, fully automated approaches entail automation of both the composition plan and the discovery of appropriate atomic web services. As the number of available services continuously increases over time, automation of the composition process constitutes the only solution able to efficiently manage the vast volume of this domain. Other advantages include scalability, flexibility to detect changes in atomic service definitions, and dynamic handling of service failure/unavailability.

Automated approaches typically involve representation of the composition problem in such a way that well-defined and long-studied AI techniques can be utilized to obtain solutions to the problem. The use of semantics can significantly facilitate the representation process and enhance employment of intelligent techniques.

Theoretical works, such as the Causal Link Matrix (CLM) [35], 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 [10] was initially created as a general-purpose, heuristic-driven Hierarchical Task Network (HTN) planning system. It was later used for automated web service composition. OWL-S process models are encoded as SHOP2 domains, while the web service composition problem is encoded as a planning problem. 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 [11]. 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 [36] 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 approach presented in [12] 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 [13] 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.

OWLS-XPlan [14] 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 middleware presented in [32] is able to transform a group of OWL-S web service descriptions into a temporal HTN domain. The composition is performed with a combination of a built-in HTN planner called SIADEX and temporal reasoning. Although this system is very interesting, as it deals with the aspect of temporality, it does not utilize the full potential of the semantic descriptions of web services. The middleware is not capable of processing semantics appropriately in order to perform relaxed matching; therefore, exact matching is required.

The motivation for our work is to propose an approach for automated service composition through planning, avoiding possible deficiencies of the aforementioned systems. This approach extensively utilizes semantics, while maintaining high efficiency, provides interoperability with current standards, and does not require any additional, domain-expert knowledge. Therefore, it demands minimum user intervention. Moreover, it finds semantically approximate solutions, according to user desires. Also, it takes into account cases of service unavailability and handles them appropriately. Finally, the approach is modular and independent from any implementation details, such as specific planning algorithms. In this way, its implementation and the acquisition of solutions can be facilitated by a number of different systems.

3 Proposed Approach

Our approach aims at automated semantic web service composition under semantic awareness, via AI planning techniques. This is achieved by transforming the web service composition problem into a planning problem. Incorporation of semantic information expands the solution space with approximate solutions. Solutions are obtained by invoking external planners in a standardized way. In order to facilitate execution, the results are eventually transformed to the original web service context.

The proposed approach achieves a high degree of automation, facilitating web service composition, even for non-experts. The overall user experience is not significantly different from the experience of discovering and invoking an atomic web service.

The key features of this approach are:

- Natural representation of web services in planning terms, due to straightforward mapping of OWL-S to PDDL elements.
- Flexibility in the selection of external planning system, due to the independence between representation and solving.
- Extensive exploitation of semantic information (semantic web service descriptions and accompanying ontologies).
- Compliance with prominent standards.
- Full-cycle composition support, as the procedure initiates from web service descriptions in OWL-S, and results in composite service descriptions in the same standard, facilitating deployment of the produced composite services.

The discrete steps constituting the overall approach are:

1. Problem transformation
2. Visual representation (optional)
3. Semantic enhancement
4. Solving through external planners
5. Composite service accuracy assessment
6. Service replacement (optional)
7. Reverse transformation

The first step is the translation of the web service composition problem into a planning problem. It concerns both the available OWL-S web services and user requirements of the composite service. It produces a planning domain and problem description, which can be consequently visualized. The next step involves semantic enhancement of this planning domain and problem. This is significant, as in many cases the syntactical differences among concepts prohibit planning systems from successfully matching inputs to outputs, even if they are semantically equivalent or very similar. The planning domains and problems derived from the initial web service composition problem, possibly semantically enhanced, are exported to PDDL so they can be solved by external, PDDL-compliant, domain independent planners, employing classical planning techniques. The produced plans constitute descriptions of the desired composite service.

Semantic enhancement as well as the use of multiple external planning systems might produce more than one composite services, which are compared and assessed in the next step. The optional replacement step handles cases of service failure or unavailability by replacing atomic services taking part in a composite service. Finally, the produced results are transformed to OWL-S.

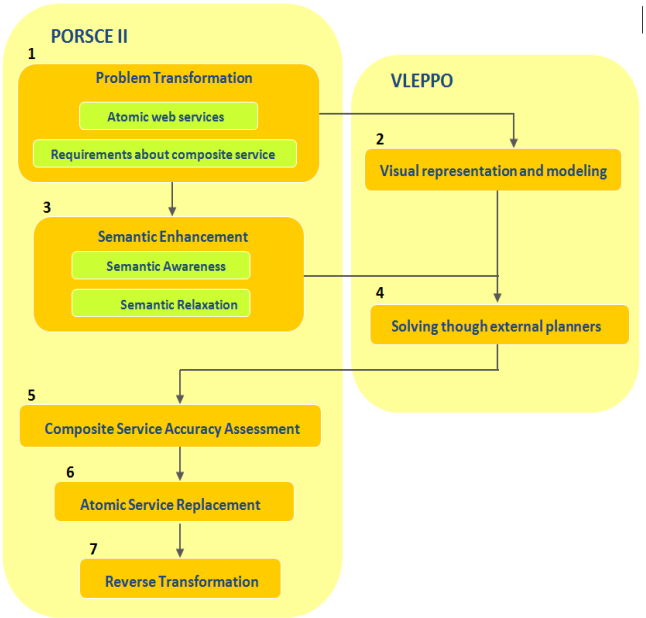

Fig. 1. Proposed approach steps.

All web service related steps, namely transformation, semantic enhancement, service accuracy assessment, service replacement and reverse transformation, are implemented in the PORSCE II system [1]. Planning related steps are accommodated by the VLEPPO system [2][24], which can also be used as a general purpose planning system for other applications. Implementation of these steps in the two systems is depicted in Fig. 1.

Although we have developed PORSCE II and VLEPPO systems to support the proposed steps, other existing systems could be used as well. In this case, the proposed methodology should be viewed as a plan depicting how these systems should be integrated.

The architecture of the two software systems implemented to support the proposed approach and the way they are integrated are depicted in Fig 2.

PORSCE II is responsible for automated composition of semantic web services through planning. It comprises of five subcomponents: OWL-S Parser, OWL Ontology Manager (OOM), Transformation Component, Visualizer and Service Replacement Component.

The OWL-S Parser adds the available OWL-S web service profiles and the corresponding ontologies in the system. OOM applies algorithms for discovering concepts that are semantically similar to a query concept, according to similarity metrics. The Transformation Component expresses the problem of web service composition as a planning problem. Moreover, it interacts with the user to set semantic similarity thresholds and enhances the planning problem with semantic information retrieved from the OOM. Additionally, it cooperates with external planning systems, which search for composition plans by matching OWL-S profile input and output parameters. When a solution to the problem is acquired, the Transformation Component uses information about concepts to assess its accuracy. Finally, it transforms the produced composite services back to OWL-S. The purpose of the Visualizer is to provide the user with a visual description of the plan representing the composite service, along with its calculated accuracy. Finally, the Service Replacement Component enables the user to modify the composite service by replacing a specific atomic web service in it.

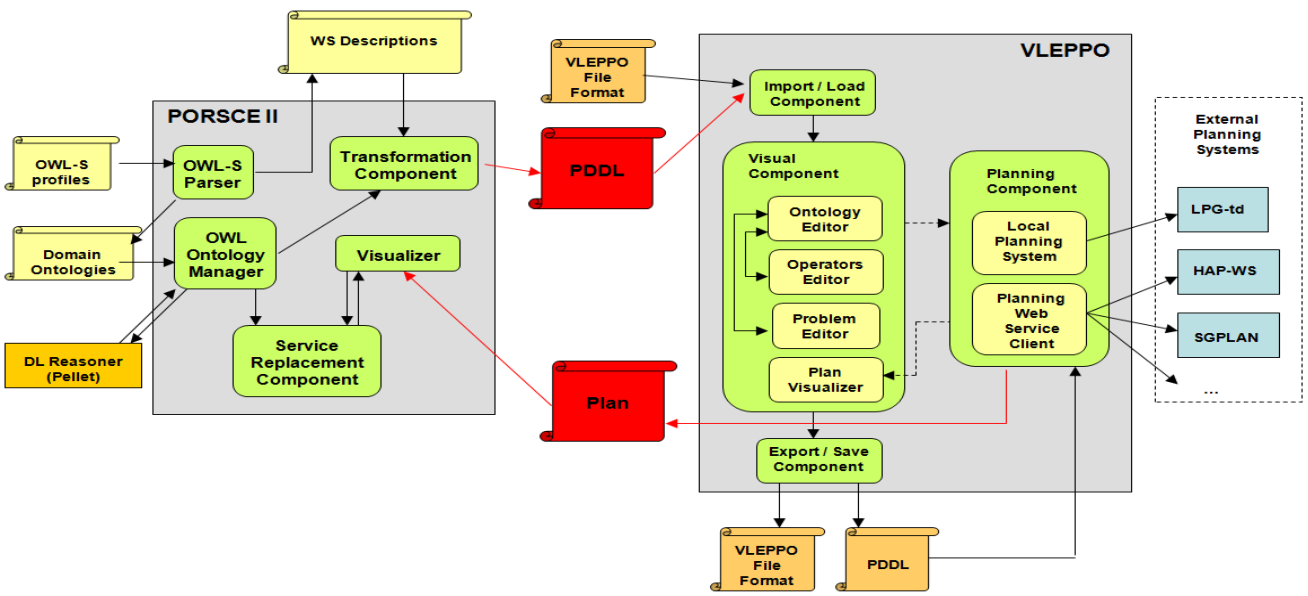

Fig. 2. Architectures of PORSCE II and VLEPPO.

Key features of PORSCE II include:

- Parsing of OWL-S web service profiles, atomic or composite, and translation into PDDL operators.
- Interaction with the user to acquire their preferences regarding the composite service and desired metrics for concept similarity.
- Enhancement of the planning domain and problem with semantically equivalent and relevant concepts by utilizing a Description Logic Reasoner.
- Acquisition of solutions to the web service composition problem.
- Composite service accuracy evaluation, based on the selected concept similarity metric.
- Visualization of solution, facilitating identification of sub-processes, such as *Sequence*, *Split* and *Split+Join* OWL-S Control constructs.
- Transformation of the solution (composite web service) to OWL-S.
- Searching through the available web services to locate semantically relevant or equivalent alternatives (atomic or sub-plan) and replacement on demand.

VLEPPO is an integrated system intended to facilitate modeling and solving of planning problems. Among its key features is a convenient and intuitive graphical interface, allowing design, comprehension and maintenance of planning domains and problems. VLEPPO maintains compatibility with standards, as most visual design elements offered in the system correspond to PDDL elements. Compliance with PDDL is also achieved through import and export features. VLEPPO offers increased flexibility in integrating planners that are exploited to acquire solutions to specific planning problems. Thus, the user is not restricted to a single planning algorithm, but has the ability to experiment with different planners. Its main components are Visual Component, Planning Component, Import / Load and Export / Save Component.

Advantages of the proposed approach include the ability to determine how to form valid composite services satisfying given goals, based only on the OWL-S descriptions of the web services and the corresponding ontologies. No prior or additional knowledge is demanded, since the ontologies capture adequately the semantics of the concepts used. Furthermore, the decision concerning the quantity and quality of the results, i.e. the number of composite services produced and their accuracy in achieving the given goals, is up to the user. Even when no exact composite services can be found, the system is able to utilize semantic information to find composite services that approximate best the desired goal. Moreover, service failure is handled through atomic service replacement, without any obligation to perform planning again, a feature not provided by similar systems. Another important aspect concerns the independence of problem representation from problem solution, thus enabling exploitation of planning research advances, instead of a built-in planner. Further advantages include conformation to current standards and increased interoperability and scalability.

4 PORSCE II

This section elaborates on the aspects of the approach accommodated by the PORSCE II system.

4.1 Semantic Analysis

OOM exploits semantic information contained in the descriptions of web services to enhance the composition procedure. For that reason, it manages the domain ontologies used to annotate the input/output parameter concepts of web service descriptions. Management of domain ontologies involves a reasoning procedure that computes the inferred ontology relationships, utilizing the Pellet DL Reasoner [16]. Consequently, concept relevance criteria are applied to determine semantically equivalent and relevant concepts to a specific query concept. In the proposed approach, two ontology concepts are considered relevant if and only if (a) they have a specific hierarchical relationship, and (b) their semantic distance does not exceed a user-defined threshold.

Hierarchical Relationships

Four possible hierarchical relationships exist between 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. $A \mid B$.
- subsume(A,B): The concept A should subsume the concept B, i.e. $B \mid A$. In both the plugin and subsume cases the subsumption relationships of equivalent concepts are not considered.
- sibling(A,B): The two concepts have a common, but not necessarily direct, superclass T, such as $A \mid T \wedge B \mid T$.

Semantic Distance

Two methods for determining the semantic distance between two ontology concepts are provided [1]:

The *Edge-Counting Distance (ec)* is based on the distance of two concepts in terms of the number of edges found on the shortest path between them in the ontology. An edge exists between two concepts A and B if A is a direct subclass of B.

The *Upwards Cotopic Distance* [17] is defined in terms of the upwards cotopic measure, denoted as uc(A), that represents the set of superclasses of concept A, including A itself. In the proposed approach, this definition has been modified to incorporate the semantics of an ontology hierarchy and it is calculated as (1):

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

In both cases, the implementation of the semantic distance metric between two concepts returns a value between 0 and 1, with 1 denoting absolute mismatch.

4.2 Problem Transformation

A typical web service composition problem involves

inputs, or data that the user is willing to provide to the composite service, and outputs, which reflect the desired results of the composite web service functionality. It also includes a number of available web services which can be combined to achieve a goal. The first step towards utilizing planning to solve a web service composition problem is translation of this problem in planning terms.

A planning problem is modeled according to STRIPS (Stanford Research Institute Planning System) notation [18] as a tuple <*I, A, G*> where *I* is the initial state, *A* is a set of available actions and *G* is a set of goals. States in STRIPS 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. The following formulae hold for the states in STRIPS notation:

- An action A_i is applicable to a state *S* if $prec(A_i) \subseteq S$.
- If A_i is applied to *S*, the successor state *S'* is calculated as $S' = S - del(A_i) \cup add(A_i)$.
- The solution to a planning problem (plan) is a sequence of actions, which, if applied to *I*, lead to a state *S'* such that $S' \supseteq G$.

The solution adopted by our approach for mapping the web service composition problem to a planning problem is the following: Let *IC* be the set of concepts that the user can provide to the composite service and *GC* its desired outputs. If *O* denotes the set of all available ontology concepts, 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 composite service formulate the goals of the problem: $I = IC$ and $G = GC$.

The available OWL-S web service descriptions are used to obtain the available actions in the planning domain. More specifically, each web service description WSD_i is translated to a domain action A_i, using the information provided by the corresponding profile instance (each web service description is actually an instance of the OWL-S Profile class). More specifically:

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

$$\text{name}(A_i) = WSD_i.\text{ID}$$

- The preconditions are based on the service input and precondition definitions (concepts):

$$prec(A_i) \equiv \bigcup_{k=1}^{n} \{WSD_i.hasInput_k\} \cup \bigcup_{k=1}^{m} \{WSD_i.has\,\text{Pr}econdition_k\}$$

- The add effects comprise of the service output and positive effect definitions (concepts):

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

- The delete list is formed by the negative effect definitions (concepts). The SWRL language [25] was used in order to model the preconditions and effects of the web services. Preconditions are modeled by SWRL rule conditions, while positive effects are modeled as SWRL atomic expressions that are true in the world after the execution of the web service. Since SWRL does not directly support negation and negated atomic expressions, which would model negative (delete) effects, the negation element of RuleML [26] was employed. The <neg> element is used by the transformation process in order to discriminate between add and delete effects. The delete list of the action is formulated as follows:

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

Invocation of planning algorithms over the newly formulated planning problem produces plans, representing the description of the desired composite web service.

4.3 Semantic Awareness and Relaxation

Successful composition is facilitated if the planning system is aware of possible similarities among syntactically different but semantically equivalent concepts. Semantic awareness enables planners to match preconditions and effects correctly, even if the terms used to refer to them in the web service OWL-S profiles differ [37].

Furthermore, in cases when no exact matching of concepts is possible, the approach is able to utilize, apart from equivalent concepts, semantically similar concepts as well. In this case, input concepts can be matched to output concepts approximately. Semantic relaxation enables the formulation of composite web services that are less accurate; nevertheless, they serve the purpose of the user in the best possible way.

For implementation of semantic awareness and relaxation, the produced planning domain and problem are enhanced with semantic information, thus maintaining planner independence. In a pre-processing step, semantically similar concepts for the facts of the initial state and the outputs of the available actions are discovered. Semantic enhancement is based on the following rules:

- The original concepts of the initial state together with the equivalent and semantically 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 including in the effects list of each operator all equivalent and semantically similar concepts for the concepts in its initial effects list.

Semantic enhancement, as described above, is performed in cycles. The thresholds for semantic similarities are increased gradually, and independent thresholds are used for each hierarchical relationship. In this way, solutions will be returned in an increasing order of relaxation, starting with exact solutions, if such exist.

4.4 Solution Acquisition and Visualization

The planning domain and problem produced in the previous steps is encoded into PDDL and solved by external planners in VLEPPO. The derived plans, which might be either sequential or partially parallel, structured

in levels, are visualized in PORSCE II. Examples of this visualization can be found in Section 6. Visualization facilitates comprehension of the structure of the produced composite service.

4.5 Composite Service Accuracy Assessment

In many cases the user is presented with multiple composite services, due to semantic relaxation and use of different planners. Despite the fact that all of these composite services cover the requested functionality, some of them may be more preferable. Therefore, statistics such as the number of actions and the number of levels in a plan, as well as accuracy metrics, such as the distance quality metric, have to be calculated for each composite service.

For the calculation of the distance quality metric, each concept appearing in the plan is annotated with a semantic distance 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 it has with the original concept (plugin, subsume or sibling relationship).

If there are a total of n concepts and each concept c_i is annotated with a semantic distance value of d_i, returned from the OOM, and a corresponding weight of w_i, depending on its hierarchical relationship to the original concept, the distance quality metric for the case of the edge-counting distance (*Plan Semantic Distance for edge-counting – PSD_{ec}*) is calculated as a weighted sum of the distances of all concepts appearing in the plan (2):

$$PSD_{ec} = \sum_{i=0}^{n} w_i d_i \quad (2)$$

When the upwards cotopic metric is used, the distance (*Plan Semantic Distance for upwards cotopic – PSD_{uc}*) is calculated as a weighted product of the all concept distances appearing in the plan, excluding equivalent concepts (3):

$$PSD_{uc} = \prod_{i=0}^{n} w_i d_i, d_i \neq 0 \quad (3)$$

The weights are used to indicate that, from a semantic perspective, in specific domains, several hierarchical relationships are more/less preferable. 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.

4.6 Atomic Service Unavailability Handling

Atomic web service unavailability occurs when attempts to access a service using the interface described in its definition are unsuccessful. In such cases, the ability to replace this service in the composite service description is essential. Service replacement can also deal with cases when the user is unwilling to use a certain web service due to lack of trust to its provider, security concerns, cost or time constraints, etc.

Service Replacement Component initiates search from a selected atomic service included in the composite service. It discovers all atomic services that could be used alternatively, and performs replacement as indicated.

Discovery of alternative atomic services requires advice from the OOM as far as equivalent and semantically similar concepts are concerned. 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. Therefore, both the following conditions must hold:

- $prec(A) \subseteq S$, where S is the state of the world exactly before the application of this action.
- $S\text{-}del(A) \cup add(A) \supseteq S'$, where S' is the set of facts that must definitely hold after the execution of A in order for the rest of the plan to be applicable. S' is given by applying Algorithm 1 to the part of the plan starting after Q.

Algorithm 1 Computes the minimum set of facts that must hold in a state S in order for a plan π to be applicable in S.
Inputs: $\pi = \{A_1,..,A_n\}$: a plan
Output: S': a set of facts

set $S' \leftarrow \{\}$
for i $\leftarrow$ n **downto** 1
  set $S' \leftarrow S' \cup \text{prec}(A_i) \setminus \text{add}(A_i)$
return S'

Note that in cases where the original plan was parallel, the above procedure should first serialize the plan, then search for possible replacements and finally attempt to recreate the possible parallel structure on the new plan. This is required in order to increase the number of possible candidates for action Q. Otherwise, any candidate replacement should not only lead to a valid plan but also maintain the same parallel structure of the initial plan.

Alternatively, if no atomic service candidates exist, Service Replacement Component can perform replanning in order to locate a set of services that have the same effect as the one being replaced, and therefore can substitute it. States S and S', as presented above, serve as initial and goal states for the replanning process, respectively.

In both cases, lists of alternatives are populated and the user is enabled to select among them. The selected alternative substitutes the original service both in the plan and in the visualization, and accuracy metric is appropriately adjusted to reflect the current composite service.

4.7 Solution Transformation

The composition process is completed by transforming the produced composite service into OWL-S. This process takes into account semantics included in OWL-S descriptions and domain ontologies. OWL-S establishes a framework for defining composite processes as a set of atomic processes, combined together using a number of control constructs, such as *Sequence*, *Split*, and *Split+Join*.

Algorithm 2 presents the basic algorithm that creates a composite service, given a web service graph. A web service graph is a graph G=(V, E), where the nodes in V correspond to all atomic services in the plan. 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 2 processes every root node in the graph and produces as output a composite construct of either the form *sequence*(c_1, c_2), or *split*(c_1, c_2,.., c_n), where c_1 to c_n are either NULL or composite constructs.

Algorithm 2 (Basic) Computes an initial composite service with *Sequence* and *Split* constructs
Inputs: G=(V,E): the web service graph
Output: C: a composite service construct

set $R \leftarrow \{r \in V: \forall x \in V, (x \rightarrow r) \notin E\}$ // *R*: set of root nodes in G
if $|R| = 0$ **then return** NULL
if $|R| = 1$ **then**
  set G'← the tree in G with r∈R as the root
  return *sequence*(r, *Basic*(G'-{r}))
set c ← {}
for each r in *R*
  set G'← the tree in G with r∈*R* as the root
  set c←c∪*Basic*(G'-{r})
return *split*(c)

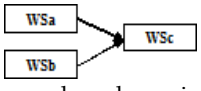

Fig. 3. Example web service graph.

For example, consider the web service graph presented in Fig. 3. The output of Algorithm 2 will be:

```
split(sequence(WSa,sequence(WSc,NULL)),
      sequence(WSb,sequence(WSc,NULL)))
```

This output is then further processed, by checking the elements of each *Split* construct in pairs. If common last arguments are located (such as the `sequence(WSc,NULL)` argument in the previous example), this suggests that a join is possible and the pair can be condensed by introducing a *Split+Join* construct. The result of this process for the above example is a construct of the form:

```
split(sequence(split+join(WSa,WSb),
      sequence(WSc,NULL)))
```

Finally, the output is simplified by removing NULLs and single-argument constructs. The final outcome for the above example composite service is the construct:

```
sequence(split+join(WSa,WSb),WSc)
```

The transformation of the solution to OWL-S facilitates deployment of the composite service in OWL-S execution systems such as the OWL-S Virtual Machine [23].

5 VLEPPO

This section overviews the functionality of the VLEPPO system pertinent to the proposed approach. Description will be confined to the features regarding web service composition; a full elaboration on the advanced features of the system is out of the scope of this paper and can be found in [24].

5.1 Visual Representation and Design

The key feature of Visual Component is its simplicity and convenience. Planning domains and problems are represented using graphical notations, bearing a high degree of correspondence to PDDL 2.2 elements. Plans, representing composite services, can also be visualized, provided that they comply with the PDDL+ standard [15].

The Domain Entities and Relationships

Planning domain structure is described by employing the formalism of the entity-relationship model, adapted to the PDDL standard. Classes are mapped to entities and predicates are mapped to relationships.

In the web services case, predicates of the planning domain represent concepts serving as inputs and outputs of services. Classes may represent arguments of these inputs and outputs, if such exist. An example of the visualization of an entity-relationship model is depicted in Fig. 4. The example concerns a partial web service domain including identification information of a certain individual, and phone number data.

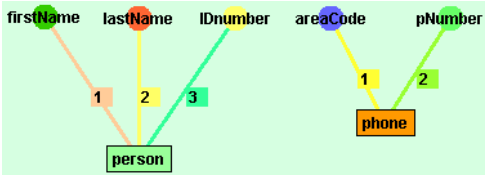

Fig. 4. Entity-relationship model example.

Representing Operators

Operators in VLEPPO have a direct correspondence to PDDL actions. The essential elements of their definition are preconditions, results (or add / delete lists), and parameters, and they are visualized in the Operators Editor.

The default view for an operator is preconditions / results view. This view depicts the preconditions that must hold for the action to be executed and the state of the world after the execution of the action (in terms of predicates affected by this action), as depicted in Fig. 5.

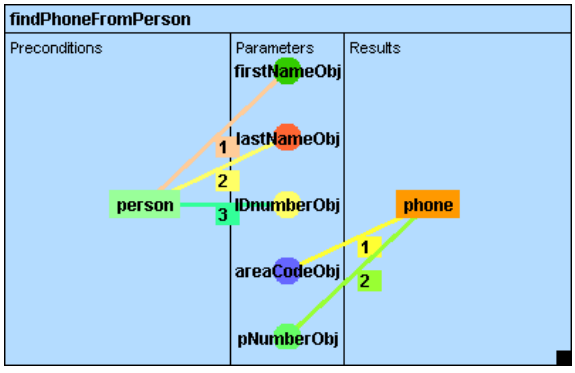

Fig. 5. Operator example.

Another option is the add / delete lists view. This view depicts the facts that will be added and deleted from the current state of the world upon the application of the action. This option is more convenient for representing inputs and outputs in the web service case. An example of this view can be found later, in the Case Study section.

Moreover, the system supports operators with duration, referred to in PDDL as *durative actions*. These can be

used instead of simple operators to represent web services whose descriptions provide temporal information, such as estimated response and execution time. Such descriptions occur by complementing OWL-S with additional ontologies, such as DAML-Time [33][34], which introduce temporal web service properties.

Representing Problems

For every domain defined in PDDL, a large number of corresponding problems can be defined by describing an initial and a goal state. In the web services case, the problem represents the user requirements (inputs and outputs) of composite services. Problems are represented by denoting the predicates in the initial state, the predicates in the goal state, and the objects that take part in the problem definition, as depicted in Fig. 6.

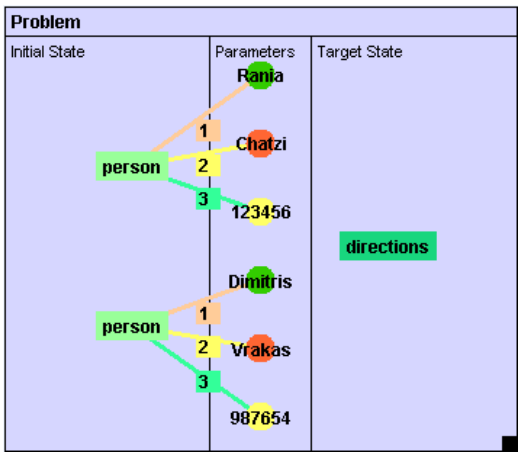

Fig. 6. Problem instance example.

Syntax and Validity Checking

A very important feature of VLEPPO is real-time syntax and validity checking, as it detects errors and inconsistencies at the time they emerge and prevents them from propagating in the domain. Planning domains are checked for consistency within their own structures, and planning problems have to be checked for consistency and correspondence to the related domains [24].

The validity checks verify the correct structure of the operators representing web services; therefore, errors in the web service definitions, such as inconsistent inputs or outputs can be detected. Moreover, the system verifies the consistency of the types of the parameters both in operators and problems. This ensures that all available web services, as well as the desired composite service, will have valid concepts as inputs and outputs. At the time of export, VLEPPO ensures that the domain and problem are complete, and all essential classes, operators and parameters have been defined, thus preventing planning errors for the external planning systems.

Syntax and validity checking can be a very useful feature in the web services case as currently there are no systems providing such functionality.

5.2 Interoperability with PORSCE II through PDDL

The feature of VLEPPO that enables interoperability with PORSCE II and other systems is the ability to import from and export to PDDL. In addition, compliance with PDDL results in increased flexibility in selecting external planning systems for acquiring solutions.

During export, elements taking part in domain definition can be combined to formulate constructs and exported to a PDDL domain file. The domain is automatically enhanced with the appropriate *requirements* tag, as detected by the system. Elements that formulate the problem are exported in a separate PDDL problem file, corresponding to a specific domain. Plans produced by external planning systems are exported in PDDL+.

Importing and visualizing planning domains and problems expressed in PDDL serves comprehension, manipulation and maintenance purposes. The designer is thus enabled to modify existing domains and problems in an intuitive way, even if they are not familiar with PDDL syntax. Both typed and non-typed PDDL files are supported; however, importing non-typed PDDL is subject to some restrictions. If no typing is used, syntax alone might not be enough, and semantic information might be necessary to discriminate types (or timeless unary predicates) from ordinary unary predicates. Most domains produced from a web service composition problem are classified in the non-typed case. To cope with that, a module for translating non-typed to typed PDDL has been developed. The module scans the non-typed PDDL file for unary predicates, which are candidates for being considered as "types". Consequently, it examines which of them could be timeless, using the distinctive property of timeless predicates that they do not appear in any add or delete lists. The non-typed to typed PDDL translation module has the best possible results, provided the information contained in PDDL files. However, when the domain is not well formed, the intervention of a user, in order to interpret semantics, is required. Even so, the effort is significantly reduced compared to designing the domain from scratch.

Load / save functions for both domains and problems can be used alternatively instead of import / export. They are capable of preserving additional visual information, such as colors and positions of elements, even for domains that are under development.

5.3 Solving

Interface with planners implemented as web services

As VLEPPO is intended to be an integrated system not only for designing but for acquiring solutions to planning problems as well, interoperability with planning systems is necessary. This is achieved by providing the ability to discover and communicate with web services offering implementations of various planning algorithms. Moreover, existing planning systems can expose their functionality through web services and be utilized by VLEPPO.

To this end, a dynamic web service client has been developed as a subsystem. In this way, the system can ex-

ploit alternative planning web services according to the problem at hand, as well as cope with changes in definitions of these web services. Problems can be solved with many planners simultaneously, without any overhead to local resources.

Communication with web services is performed by means of exchanging SOAP (Simple Object Access Protocol) messages, as the web service paradigm dictates. However, in a higher level, the communication is facilitated by the use of the PDDL language, which constitutes the common ground between VLEPPO and the planners. An additional advantage of using the PDDL standard is that the system is not obliged to determine which PDDL features the planners can handle, thus leaving each planning system to decide for itself.

Employment of the web service technology results in the independence of the approach from the planning or problem solving module and increased flexibility. Such a decoupling is essential since new planning systems that outperform current ones are being developed. Each of them can be exposed as a web service and then invoked for solving a planning problem without any further changes to the domains and problems already designed and exported as PDDL files.

Solving planning problems locally

Although VLEPPO aims at exploiting the capabilities of web service technology to take advantage of different planners, according to the problem at hand, an option to solve the problems locally is also offered. This option can be used at any time without any special machine set up. Therefore, the lack of internet connectivity or planning web services does not prevent users from obtaining valid compositions, although resulting compositions in this case might not be optimal. Currently, the planners used for solving problems locally are LPG-*td* [20], which proved to perform very well based on the results of International Planning Competitions, and JPlan [22], which is an open-source implementation of Graphplan.

6 Case Study and Performance Results

This section aims at demonstrating the application of the proposed approach, following the course of Fig. 1. Additionally, it intends to evaluate performance and scalability of the approach for large numbers of available web services. A major goal is to emphasize compatibility with existing web service test sets and ontologies. The test sets used to perform experiments were obtained from the SemWebCentral OWLS-TC version 2.2 revision 1 [21]. They included web services classified in various domains such as books, economy, food and travel, accompanied by corresponding ontologies. Several additional service descriptions of interest, which were included to illustrate the full capabilities of the system, appear in Table 1. For experiments, the entire set of web services included in specific domains is taken into account, so that the produced domain size is maintained in realistic levels.

Translation of all available OWL-S atomic web services for this case study is performed in the Transformation Component of PORSCE II. Each atomic web service is transformed into a planning action, with the service inputs represented as preconditions and the service outputs mapped to results. Consequently, if the user wishes to, the produced domain can be imported to VLEPPO for visualization purposes. The visual representation for some of the actions of this case study is depicted in Fig. 7.

Table 1. Add / modified web services.

Service	Inputs	Outputs
BookToPublisher	Book, Author	Publisher
CreditCardCharge	OrderData, CreditCard	Payment
ElectronicOrder	Electronic	OrderData
PublisherElectro-nicOrder	PublisherInfo	OrderData
ElectronicOrde-rInfo	Electronic	OrderInformation
Shipping	Address, OrderData	ShippingDate
WaysOfOrder	Publisher	Electronic
CustomsCost	Publisher, OrderData	CustomsCost

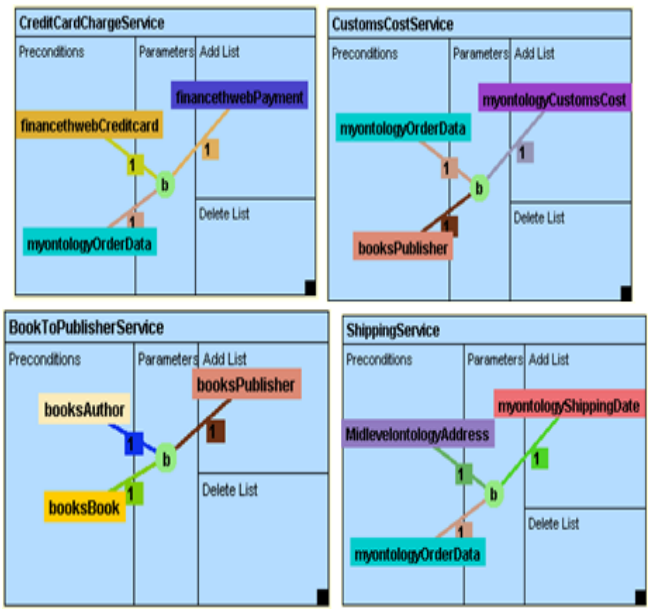

Fig. 7. E-bookstore domain operators.

The scenario implemented here concerns the electronic purchase of a book. The user wishes to provide as inputs a book title (books.owl#Book) and author (books.owl #Author), credit card information (finance_th_web.owl #credit_card) and the address that the book will be shipped to (Mid-level-ontology.owl#Address). The outputs of the desired composite service are a payment from the credit card for the purchase (finance_th_web.owl #payment), as well as shipping dates (my_ontology.owl #ShippingDate) and customs cost (my_ontology.owl #CustomsCost) for the specific item.

User requirements for the desired composite service are expressed either in PORSCE II through a dialog interface such as the one depicted in Fig. 8, or visually in VLEPPO as a planning problem, as shown in Fig. 9.

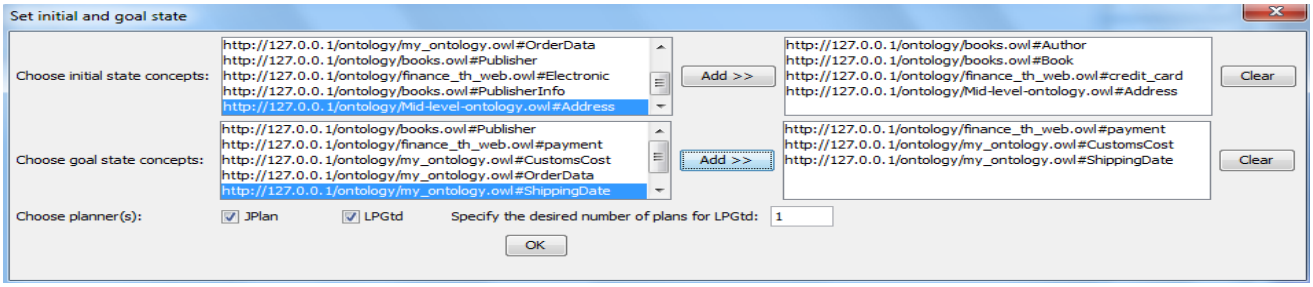

Fig. 8. Initial and goal state definition in PORSCE II.

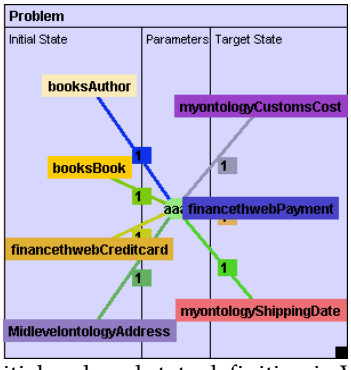

Fig. 9. Initial and goal state definition in VLEPPO.

The next step is optional semantic relaxation, performed through semantic enhancement. While exact matching of input to output concepts is obligatory in classical planning domains, this might not be the case for the web services world. Generally, it is considered preferable to present a composite service that approximates the required functionality than to present no service at all. In such cases, semantic relaxation can be proved very useful. The semantically relevant and equivalent concepts needed for implementing semantic relaxation are obtained from the OOM, while the user has control over the degree of relaxation by defining semantic distance metrics, hierarchical relationships and thresholds.

At this point, PORSCE II exports the formulated (and possibly semantically enhanced) planning domain and problem to PDDL. The full domain and problem for this case study, visualized in VLEPPO, is presented in Fig. 10.

In order to acquire solutions, both domain and problem are imported in VLEPPO, which for this example invokes LPG-*td* locally, using the operator set described above, without including any semantically relevant concepts. The result is presented in Fig. 11.

If an exact matching service is impossible to be found, then the user might resort to approximate services through semantic relaxation. Such an approximate service for the specific case study is presented in Fig. 12. The calculated accuracy of this approximate service is different from the accurate one presented in Fig. 11.

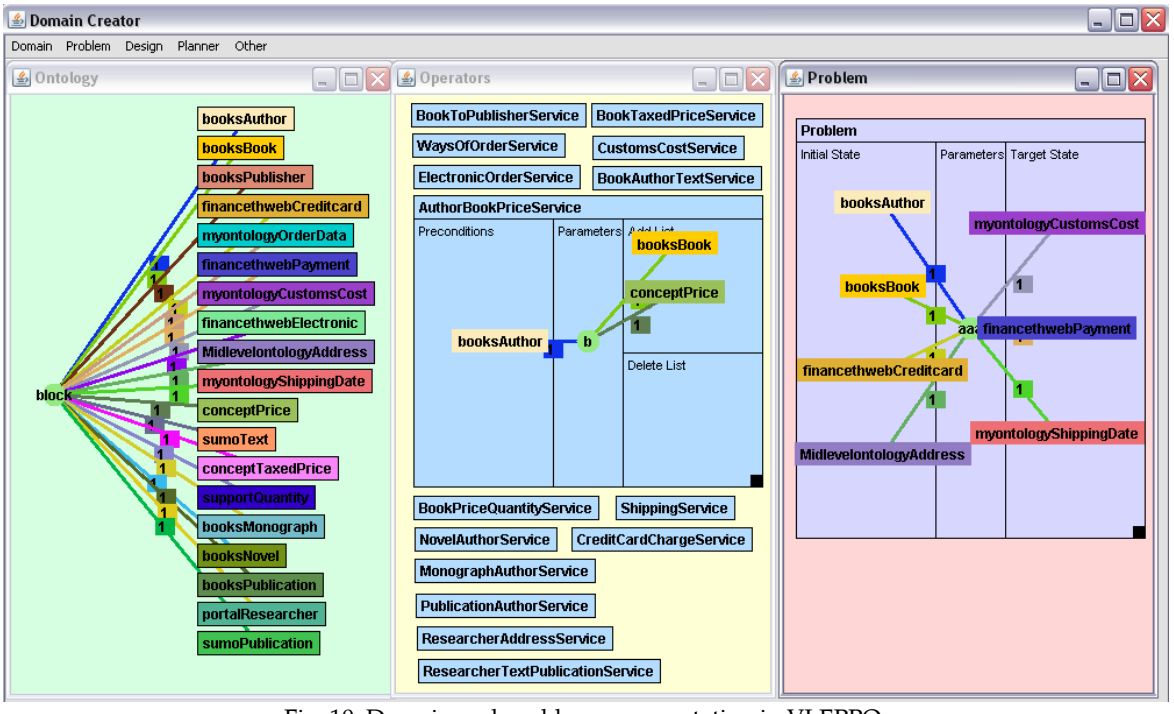

Fig. 10. Domain and problem representation in VLEPPO.

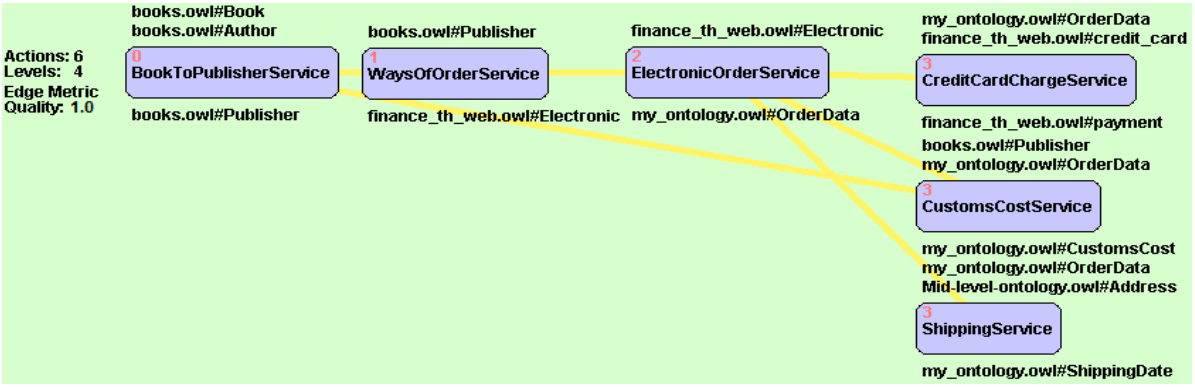

Fig. 11. Accurate plan. As no relaxed matching is performed, accuracy quality is the best possible.

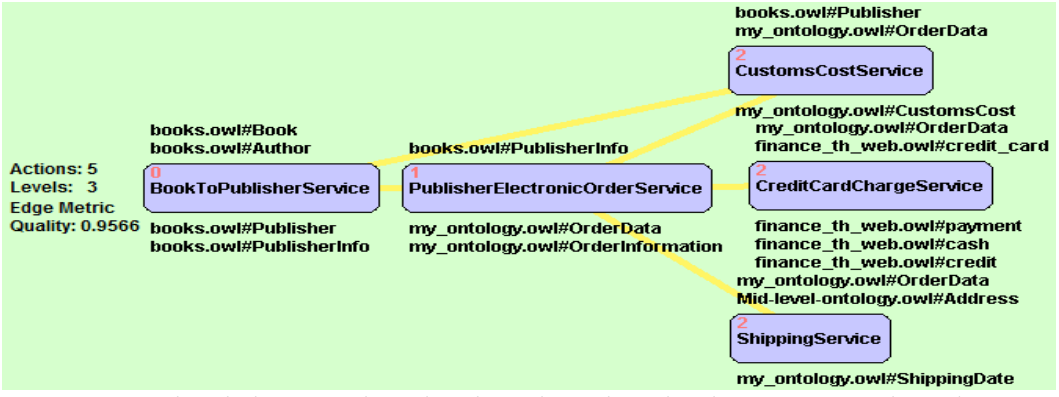

Fig. 12. Relaxed plan. Note that relaxed matching alters the plan accuracy quality value.

Experiments performed intended to study the behavior of the system as the number of available web services increases significantly. Scalability issues are emphasized as they have not been previously explored for other related systems. For example, OWLS-XPlan has only been studied for domains up to approximately 50 services, and no performance measures are provided. The ability of a web service composition system to scale up and maintain efficiency is very important, as it is one of the main factors that determine the applicability of the approach in real world domains, with hundreds of web services.

Table 2. Time measurements (in milliseconds)

Number of web services		10	100	500	1000
Preprocessing time (PORSCE II)		5857	6104	5875	5703
Total transformation time (PORSCE II)	X	4594	70062	350836	792109
	E	4531	75725	335477	796797
	C	4585	74688	728633	3901141
Transformation time per WS (PORSCE II)	X	459	700	702	792
	E	453	671	757	797
	C	459	746	1457	3901
Planning time (LPG-*td*) (VLEPPO)	X	1	13	16	17
	E	4	6	15	16
	C	3	5	16	16

In experiments, 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, OWL-S to PDDL transformation 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).

Preprocessing times did not show significant fluctuation, as they depend only on the number and structure of processed ontologies and not on the number of available web services. Preprocessing time is not negligible; however, this process is done offline, only once. Then, results are taken into account for a large number of user requests. Preprocessing does not need to be performed again as far as the domain ontologies are not significantly altered.

As far as the scalability of the system is concerned, total transformation time evidently increased as the number of available web services increased. However, the calculation of the average transformation time per web service profile shows that it converged to approximately 0.8 seconds for the exact matching and the edge-counting distance metric cases. For these cases, the complexity of the transformation process is linear. In the upwards cotopic metric distance case, the increase in the average transformation time appears to be significant as available web services increase. This overhead is introduced during the location of semantically similar concepts, in order to perform semantic relaxation. The delay happens due to higher complexity of the algorithm used for the calcula-

tion of the upwards cotopic relevance between concepts, compared to the edge-counting case.

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, as it uses graph structures to exclude unrelated operators early during planning. Semantic relaxation does not impose additional overhead to planning, as it does not increase the number of operators. Planning time is not the most important factor that affects system performance, as no specific planner is inherent in the proposed system.

Table 3 shows the increase in the number of plans as semantic distance thresholds for different hierarchical relationships increase during semantic relaxation iterations. The quality metric values for the produced solutions are also depicted.

Table 3. Number of plans and accuracy metric values.

Thresholds	sup:0 sub:0 sib:0	sup:0 sub:0 sib:1	sup:1 sub:0 sib:1	sup:1 sub:1 sib:0	sup:1 sub:1 sib:1
# of plans	2	2	3	4	4
Plan accuracy metric values	1 1	1 1	1 1 0.9783	1 1 0.9783 0.9566	1 1 0.9783 0.9566

A comparison with respect to semantic relaxation techniques (edge-counting or upwards cotopic) would not produce meaningful conclusions, as the returned results highly depend on the structure of each ontology. Instead, semantic relaxation techniques should be viewed as alternative and complementary ways to retrieve semantically related concepts to the ones of interest.

7 Conclusions and Future Work

This paper proposes an approach utilizing AI techniques to address automated web service composition, which has emerged as a significant issue in the research community. The automation of the composition procedure is essential, as it permits handling of the continuously increasing numbers of available atomic web services. A significant contribution of the proposed approach concerns the full incorporation of semantics. The utilization of semantic information facilitates discovery and composition. It also permits approximate composition and enables the quality assessment of the produced composite services in terms of accuracy. The framework maintains compatibility with the current standards, to ensure interoperability, and it is independent from specific planners.

In the proposed approach, a web service composition problem is mapped into a planning problem. Knowledge contributed by domain ontologies is exploited to semantically enhance the produced problem, allowing approximate compositions. Solutions can be obtained by utilizing external planning systems. The produced plans, representing descriptions of the desired composite web service, are assessed in terms of accuracy. Service failures are handled by replacement of any service with an equivalent or a similar one. Finally, the produced composite service is expressed in OWL-S to facilitate its deployment. Implementation of approach was accommodated by the development of the PORSCE II and VLEPPO systems.

Future goals include the addition of the OWL-S descriptions of produced composite services in the registry of available services, to explore the possibility to accelerate the composition process. Moreover, it lies in our immediate plans to study ways to enhance the approach with the ability to produce various composite services according to non-functional user preferences, dealing with pragmatic knowledge. As web service standards evolve, exploitation of pragmatic knowledge could be possible by extending existing web service description.

References

[1] Hatzi, O., Meditskos, G., Vrakas, D., Bassiliades, N., Anagnostopoulos, D., Vlahavas, I., 2009b. Semantic Web Service Composition using Planning and Ontology Concept Relevance with PORSCE II, The IEEE / WIC / ACM Conference on Web Intelligence 2009.

[2] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, I. Vlahavas, "VLEPPO: A Visual Language for Problem Representation", PlanSIG 07, Roman Bartak (Ed.), pp. 60 - 66, Prague, Czech Republic, 2007.

[3] Srivastava, B. and Koehler, J., 2003. Web Service Composition - Current Solutions and Open Problems. ICAPS 2003 Workshop on Planning for Web Services.

[4] Milanovic, N., & Malek, M., 2004. Current solutions for Web service composition. IEEE Internet Computing, 8(6), 51 – 59

[5] Bucchiarone, A. and Gnesi, S., 2006. A Survey on Service Composition Languages and Models, WsMaTe 2006.

[6] Rao, J. and X. Su, 2004. A Survey of Automated Web Service Composition Methods. LNCScience, Vol. 3387/2005, Springer, p. 43-54.

[7] Dustdar, S. and Schreiner, W., 2005. A survey on web services composition, Int. J. Web and Grid Services, Vol. 1, No. 1, pp.1–30.

[8] Thatte S. (ed.), 2003. BPEL4WS (Version 1.1), http://www.ibm.com/developerworks/library/specification/ws-bpel/

[9] Casati, F., Ilnicki, S., Jin, L., et al, 2000. Adaptive and Dynamic Service Composition in eFlow. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, Springer, Heidelberg.

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

[11] Pistore, M., Marconi, A., Bertoli, P. and Traverso, P., 2005. Automated Composition of Web Services by Planning at the Knowledge Level, in proc. of IJCAI 05, Edinburgh, UK.

[12] McIlraith, S. and Son, T., 2002. Adapting Golog for Composition of Semantic Web Services, KR2002, pp 482-493.

[13] Ponnekanti, S.R. and Fox, A., 2002. SWORD: A Developer Toolkit for Web Service Composition, WWW 2002, Elsevier, pp. 83-107.

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

[15] Fox, M. & Long, D., 2002. PDDL+: Modeling continuous time dependent effects. In Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space.

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

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

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

[19] Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D., 1998. PDDL -- the Planning Domain Definition Language. Technical report, Yale University.

[20] Gerevini, A., Saetti, A., Serina, I., 2004. LPG-*td*: a Fully Automated Planner for PDDL2.2 Domains, in IPC, 14th ICAPS, 2004.

[21] OWLS-TC, http://projects.semwebcentral.org/ projects/owls-tc/

[22] JPlan, http://sourceforge.net/projects/jplan
[23] M. Paolucci, A. Ankolekar, N. Srinivasan and K. Sycara, "The DAML-S Virtual Machine", ISWC, 2003, pp 290-305.
[24] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, I. Vlahavas, "A Visual Programming System for Automated Problem Solving", Expert Systems With Applications, Elsevier, Vol. 37 (6), pp. 4611-4625, 2010.
[25] SWRL, http://www.w3.org/Submission/SWRL/
[26] RuleML, http://ruleml.org/
[27] Aalst, W.M.P van der, 2003. Don't Go with the Flow: Web Services Composition Standards Exposed, IEEE Intelligent Systems, Vol. 18, No. 1, pp. 72-76.
[28] Berardi, D., Calvanese, D., De Giacomo, G. and Mecella, M., 2005. Automatic composition of process-based web services: a challenge. In Proc. of the WWW'05 Workshop on Web Service Semantics: Towards Dynamic Business Integration (WSS 2005)
[29] Medjahed, B., Bouguettaya, A., 2003. Elmagarmid, A.K.: Composing Web services on the Semantic Web. VLDB J. 333-351
[30] Carman M., Serafini L., Traverso P., 2003. Web Service Composition as Planning, ICAPS 2003 Workshop on Planning for Web Services.
[31] OWL-S 1.1. http://www.daml.org/services/owl-s/1.1/
[32] J. Fernandez Olivares, T. Garzón, L. Castillo Vidal, Ó. García Pérez, F. Palao. A Middleware for the automated composition and invocation of semantic web services based on HTN planning techniques. CAEPIA07. Springer LNAI 4788, 2007.
[33] Hobbs, J. R., 2002. DAML-Time. 'A DAML ontology of time', http://www.cs.rochester.edu/_ferguson/daml/daml-time-20020830.txt.
[34] Pan, F.; Hobbs, J. R. 'Time in OWL-S'. In Proceedings of AAAI-04 Spring Symposium on Semantic Web Services, 2004.
[35] F. Lecue, A. Leger: A Formal Model for Semantic Web Service Composition. ISWC 2006: 385-398.
[36] McDermott, D., 2002. Estimated-regression planning for interactions with web services. ICAPS'02.
[37] M. Paolucci, T. Kawmura, T. Payne, K. Sycara, Semantic Matching of Web Services Capabilities, First International Semantic Web Conference, 2002.
[38] Aalst, W.M.P. van der, Dumas, M. and ter Hofstede, A.H.M., 2003. Web Service Composition Languages: Old Wine in new Bottles. In: 29th EUROMICRO Conference.
[39] Alonso, G., Casati, F., Kuno, H. and Machiraju, V., 2004. Web Services. Concepts, Architectures and Applications, Springer-Verlag.
[40] Chan, M., Bishop, J., & Baresi, L., 2007. Survey and Comparison of Planning Techniques for Web Services Composition. University of Pretoria, Technical Report.
[41] Zhan, R., Arpinar, B., & Aleman-Meza, B., 2003. Automatic composition of semantic web services. ICWS'03.
[42] Lecue, F., & Delteil, A. (2007). Making the difference in semantic web service composition. In 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07).
[43] Srivastava, B., 2002. Automatic web services composition using planning. KBCS'02, pp. 467–47.
[44] Chun, S. A , Lee, Y., Geller, J. Ontological and pragmatic knowledge management for web service composition. 9th International Conference on Database Systems for Advanced Applications, 2004.
[45] Schoop, M., de Moor, A. and J. Dietz, "The pragmatic web: a manifesto", Communications of the ACM, vol.49 no.5, 2006, pp. 75-76
[46] Benfell,A., Liu, K., Specifying a Pragmatic Web-browser for the Automated Discovery of Web Services in a Service Oriented Architecture Context, 10th International Conference on Organisational Semiotics, pp 99, 2007.
[47] Pistore,M., Barbon,F., Bertoli, P., Shaparau, D., Traverso, P. Planning and Monitoring Web Service Composition, ICAPS-2004 Workshop on Planning and Scheduling for Web and Grid Services.
[48] Martinez, E., Lesperance, Y., 2004. Web service composition as a planning task: Experiments using knowledge-based planning, ICAPS-2004 Workshop on Planning and Scheduling for Web and Grid Services, pp. 62-69.
[49] Uwe Keller, Rubén Lara, Holger Lausen, Axel Polleres, and Dieter Fensel. Automatic location of services. In Proceedings of the 2nd European Semantic Web Conference (ESWC2005), May 2005.

AUTHORS' BIOGRAPHIES

Ourania Hatzi received a BSc Degree in Computer Science from the Department of Informatics, Aristotle University of Thessaloniki, Greece, in 2004, and an MSc Diploma in Advanced Information Systems from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece, in 2008. She received her PhD Degree in Semantic Web Service Composition through AI Planning from the Department of Informatics and Telematics, Harokopio University of Athens, Greece, in 2009. She is currently a research associate and an adjunct lecturer at the same department. Her research interests include the Semantic Web, Intelligent Systems and AI Planning. http://www.dit.hua.gr/~raniah/

Dimitris Vrakas is currently a Lecturer at the Department of Informatics at the Aristotle University of Thessaloniki, Greece. He received his PhD degree in Intelligent Planning Systems from the same department in 2004. His research interests include artificial intelligence, intelligent tutoring systems, Automated Planning, Heuristic Search and Problem Solving. He has published more than 30 papers in journals, conferences, and books, co-edited 2 international books in the above areas. He has been involved in projects concerning educational software, intelligent agents, e-learning, web services etc. http://lpis.csd.auth.gr/vrakas

Mara Nikolaidou is an Associate Professor in the Department of Informatics and Telematics at Harokopio University of Athens. She holds a PhD and Bachelor degree on Computer Science from Department of Informatics and Telecommunications at University of Athens. Her research interests include software and information system engineering, service-oriented architectures, e-government and digital libraries. Over the last years she actively participated in numerous projects on service-oriented architectures, digital libraries and e-government. She has published more than 100 papers in international journals and conferences. http://www.dit.hua.gr/~mara/

Nick Bassiliades received his PhD degree in parallel knowledge base systems from the Department of Informatics at the Aristotle University of Thessaloniki, Greece, in 1998. He is currently an Assistant Professor in the same department. His research interests include knowledge-based systems, rule systems, agents, and the semantic Web. He has published more than 100 papers in journals, conferences, and books, and has coauthored two books. He was on the Program Committee of more than 45 and on the Organizational Committee of 5 conferences / workshops. He has been involved in several projects concerning knowledge based systems, intelligent agents, e-learning, web services, semantic web, rules, ontologies, etc., leading 7 of them. He is a member of the Board of the Greek Artificial Intelligence Society, a director of RuleML, Inc., and also a member of the Greek Computer Society, the IEEE, and the ACM. http://lpis.csd.auth.gr/people/nbassili/

Dimosthenis Anagnostopoulos is a Professor in the Department of Informatics and Telematics at Harokopio University of Athens. He holds a PhD and Bachelor degree on Computer Science from Department of Informatics and Telecommunications at University of Athens. He has published more than 100 papers in international journals and conferences. His research interests include discrete event simulation, faster-than-real-time simulation, modeling and simulation of distributed information systems. He has actively participated in numerous projects related to simulation, e-government and information systems. http://www.dit.hua.gr/~dimosthe/

Ioannis Vlahavas is a Professor at the Department of Informatics at the Aristotle University of Thessaloniki. He received his Ph.D. degree in Logic Programming Systems from the same University in 1988. He specializes in logic programming, knowledge based and AI systems and he has published over 200 papers and book chapters, and co-authored 8 books in these areas. He teaches logic programming, AI, expert systems, and DSS. He has been involved in more than 27 research and development projects, leading most of them. He is leading the Logic Programming and Intelligent Systems Group (LPIS Group, lpis.csd.auth.gr). http://www.csd.auth.gr/~vlahavas