ABSTRACT
A problem that has become one of the recent critical issues in today’s Web environments is automated composition of Semantic Web services. A number of approaches have been proposed to tackle the problem. However, discovery and composition of Semantic Web services have been overlooked by majority of those approaches. In this paper, we propose an effective approach called AIMO, based on HTN-planning and Web Service Modeling Ontology (WSMO), which are adapted and extended to tackle the mentioned problem. Moreover, we propose a translator to provide interaction between WSMO and HTN-planning which is based on Description Logic. In addition, we have implemented some parts of the AIMO. Finally, an experimental validation has been done using a case study, in order to evaluate the efficiency and effectiveness of the proposed approach. Based on these experiments, and the implementation, the proposed approach is useable and performed.

Categories and Subject Descriptors
D.2.13 [Software Engineering]: Software Reuse-Reuse models
d.2.11 [Software Engineering]: Software Architectures-Languages.

General Terms

Keywords
Semantic Web Services, Web service discovery, Web service composition, HTN-planning, Description Logic.

1. INTRODUCTION
A Web service [1] is a piece of XML-based software interface that can be invoked over the Internet and can be roughly viewed as a next-generation successor of the Common Object Request Broker Architecture (CORBA) [2] or the Remote Procedure Call (RPC) [26] technique. The main benefits of Web services are interoperability, ease of use, reusability and ubiquitous computing.

Currently, an increasing number of companies and organizations implement their applications over Internet. Thus, the ability to select and integrate inter-organizational and heterogeneous services on the Web efficiently and effectively at runtime is an important step towards the development of the Web service applications. Recent researches study how to specify (in a formal and expressive enough language), discover and ensure the correctness of Web services. In this paper, a significant portion of the work has been dedicated to scenarios aimed at both automating Web service discovery and composition functionality.

As W3C defined [23], “Discovery is the act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria”. Indeed, Web service discovery is the process of finding a suitable Web service for a given task. When no atomic Web service can fulfill the user’s requirements, there should be a possibility to combine existing services together in order to satisfy the request requirement. This trend has inaugurated a considerable number of research efforts on the Web service composition (WSC) both in academia and industry.

Most of current approaches related to WSC applied following techniques: HTN [3], Golog [4], classic AI planning [5], rule-based planning [6], model checking [7], theorem proving [8], etc. Some approaches need too much human effort; some overlook the problem of discovery. Overcoming both discovery and composition of services is the key to automatic generation of executable process.

In this paper, an automated approach is proposed, to do both Web service discovery, based on Description Logics [11]. Firstly, the definitions of Web service discovery and composition are presented based on Description Logics. After that, architecture of the proposed approach and all its components are described. Then, in order to automatically generate the control flow of the planning process, an extension of HTN-DL formalism described in [10] that combines Hierarchical Task Networks and Description Logics is proposed. This extension involves the capability of Web service discovery using WSMO [9]. On the other hand, the proposed approach can solve the problem of WSC for WSMO using AI-planning (i.e., HTN-DL). This extension enhances the performance of HTN-DL and correctness verifiability of WSMO. Moreover, a translator is proposed to provide interaction between WSMO and AI-planning.

The remainder of this paper is organized as follows. First of all, some background information of related ideas is provided. After that, the motivation and methodology of this research are given respectively. The next section forms the technical core of this paper and details the evaluation of current WSC approaches, the proposed architecture, implementation and also experimental results. Section 6, discusses some related work in the area of WSC and differences between the proposed approach and existing...
approaches. Finally, conclusions and future work are given in section 7.

2. BACKGROUND

In this section, the concepts which are related to the proposed approach, i.e., HTN-planning, Description Logics, and WSMO are described.

2.1 HTN-Planning

HTN planner is used to produce a sequence of actions that perform some activity or task. HTN-style domains fit in well with the loosely coupled nature of Web services. This means different decompositions of a task are independent so the designer of a method does not have to have close knowledge of how the further decompositions will go or how prior decompositions occurred. The planning algorithm is a forward-decomposition HTN planning. In this algorithm, a planning problem is a triple (S, w, D) where S is the initial state, w is the task network to plan for, and D is the planning domain. The description of a planning domain includes a set of planning operators, a set of methods, and also a DL ontology called task ontology (T\textsubscript{ont}) where task symbols are used as concept names and operator and method names are used as individual names. HTN planning keeps a state of the world and uses actions to represent state transitions as classical planning does. Nevertheless, HTN planners differ from classical planners in what they plan for and how they plan for it.

A forward-search HTN planner starts with the initial task network and chooses the task that has no predecessors. For primitive tasks, an applicable operator is found and added to the plan. If task is non-primitive, an applicable method is found and the selected task is replaced with the subtasks of the method. Planning proceeds by decomposing tasks recursively into smaller subtasks, until there are no tasks left in the network. For details on HTN planning, see [12].

Nevertheless, HTN planning has been proved promising for Web service composition [13], previous work has some limitations in their use of HTN planning directly for web services described in OWL-S because of the following reasons:

- In HTN planning there is no way to infer relations between different tasks and a task is identified only by its name and the number of arguments it has.
- There is a restriction on primitive and non-primitive tasks such that it is impossible to have both an atomic service and a composite service achieving the same goal.
- There is a one-to-one mapping between operators and primitive tasks such that only one operator can accomplish any given primitive task. However, this is not true in web services; there are many different web services that can accomplish the same primitive task.

HTN-DL [10] is a planning formalism that combines HTN planning with OWL and Description Logic (DL) [11] representation. The DL is used to describe both actions and states with an expressive knowledge representation language. The service categorization and non-functional attributes of services are described in a task ontology that allows flexible matchmaking. The state of the world is also represented as a DL knowledge base. HTN-DL also differentiates between world-altering effects and knowledge effects making it possible to interleave planning with execution by invoking information-providing services during composition.

HTN-DL translates atomic services in OWL-S descriptions to HTN-DL operators. An HTN-DL operator is described as (N, I, O, P, E\textsubscript{w}, E\textsubscript{k}) where N is the operator name, I is the set of input parameters, O is the set of output parameters, P is the preconditions, E\textsubscript{w} is the world-altering effects and E\textsubscript{k} is the knowledge effects. Similarly, composite services in OWL-S are translated to HTN-DL methods. A method is an expression of the form (N, I, O, P, V, E\textsubscript{w}, E\textsubscript{k}, Γ) where V is the set of local variables in conditions and Γ is a conditional task network stating that under different conditions, there are different ways of decomposition. A task can be matched with operators or methods. If an operator is matched, the planner will execute the action and apply effects. If a method is matched, the planner continues decomposition until atomic action is reached.

2.2 Description Logic

Description Logics (DLs) [11] are a family of knowledge representation formalisms that are able to represent the structural knowledge of an application domain through a knowledge base including a terminology and a world description. The basic formalism of a DL system comprises three components: 1) Constructors which represent concept and role, 2) Knowledge base (KB) which consists of the TBox(terminology) and the ABox(world description). The TBox presents the vocabulary of an application domain, while the ABox includes assertions about named individuals in terms of this vocabulary, 3) Inferences which are reasoning mechanisms of Tbox and Abox. In the rest of the section, the syntax and semantics of the Description Logic SHIQ(D)[28] which is the DL underlying WSML-DL [27] are described briefly.

Let N\textsubscript{R} be the set of role names. The set of SHIQ\textsubscript{R} roles is the set N\textsubscript{R} ∪{R | R ∈ N\textsubscript{R}}. For R ∈ N\textsubscript{R}, let Inv(R) denote R\textsuperscript{−} and let Inv(R\textsuperscript{−}) denote R. An RBox R over NR is a finite set of transitivity axioms Trans(R) and role inclusion axioms R\subseteq S, where R and S are roles, such that, if R\subseteq S\subseteq R, then Inv(R\subseteq S)\subseteq Inv(R)\subseteq S. Let R\subseteq S denote the reflexive-transitive closure of \subseteq. A role R is transitive if Trans(S)\subseteq R or Trans(Inv(S))\subseteq R for some S with S\subseteq R and R\subseteq S; R is simple if there is no role S such that S\subseteq R and S is transitive; R is complex if it is not simple.

Let N\textsubscript{C} be a set of atomic concept names. The set of SHIQ\textsubscript{C} concepts over N\textsubscript{C} and N\textsubscript{R} is defined inductively as the smallest set for which the following holds: ⊤ and ⊥ are SHIQ\textsubscript{C} concepts, each atomic concept name A ∈ N\textsubscript{C} is a SHIQ\textsubscript{C} concept, if C and D are SHIQ\textsubscript{C} concepts and R is a role, then ¬C, C ∩ D, C ∪ D, ∃R.C, ∀R.C are also SHIQ\textsubscript{C} concepts, and, if C is a SHIQ\textsubscript{C} concept, R a simple role and n an integer, then ≤\textsubscript{n}R.C and ≥\textsubscript{n}R.C are SHIQ\textsubscript{C} concepts.

TBox T over N\textsubscript{C} and R is a finite set of concept inclusion axioms C\subseteq D or concept equivalence axioms C ≡ D, where C and D are SHIQ\textsubscript{C} concepts. For example, suppose that Person is an atomic concept and hasChild is an atomic role, using the bottom concept
the concept Person\(\sqcap\)\(\text{hasChild}\)\(\bot\), denoting those persons without a child can be formed.

Let \(N\) be a set of individual names. An ABox \(A\) is a set of concept and role membership axioms \(C(a)\) and \(R(a, b)\), and (in)equality axioms \(a \approx b\) and \(a \not\approx b\), where \(C\) is a \(\text{SHIQ}\) concept, \(R\) a role, and \(a\) and \(b\) are individuals.

A \(\text{SHIQ}\) knowledge base \(KB\) is a triple of the form \((KB_C, KB_R, KB_A)\), where \(KB_C\) is an RBox, \(KB_R\) is a TBox, and \(KB_A\) is an ABox.

The semantics of a \(\text{SHIQ}\) knowledge base \(KB\) is given by the mapping \(\pi\) which transforms \(KB\) axioms into a set of first-order formulae which can be seen in [31]. Two concepts \(C\), \(D\) are equivalent, and write \(C \equiv D\), if and only if \(\pi(C \equiv D)\). Moreover, \(KB\) is satisfiable if \(\pi(KB)\) is satisfiable.

Regarding a knowledge base \(\mathcal{T}, \mathcal{A}\), several types of reasoning problems are supported by a DL reasoning system. In this section, concept satisfiability and subsumption are introduced as follows:

- **Concept satisfiability.** A concept \(C\) is satisfiable with respect to \(KB\) if there exists a model of \(KB\) in which the interpretation of \(C\) is not empty. This is the case if and only if \(KB; C(\alpha)\) is satisfiable, where \(\alpha\) denotes a new individual not occurring in \(KB\).

- **Concept Subsumption.** A concept \(C\) is subsumed by a concept \(D\) with respect to \(KB\) if \(\pi(KB) \models \pi(C \subseteq D)\). This is the case if and only if \(KB; (C \supseteq \neg D)(\alpha)\) is unsatisfiable.

### 2.3 Web Service Modeling Ontology (WSMO)

**WSMO** defines a model to describe Semantic Web services, based on the conceptual design set up in the Web Service Modeling Framework WSMF [14]. WSMO identifies four top-level elements as the main concepts [15]:

- **Ontologies:** provide the (domain specific) terminologies used and is the key element for the success of Semantic Web services. Furthermore, they use formal semantics to connect machine and human terminologies.

- **Web services:** are computational entities that provide some value in a certain domain. The WSMO Web service element is defined as follows:
  - **Capability:** This element describes the functionality offered by a given service.
  - **Interface:** This element describes how the capability of a service can be satisfied. The Web service interface principally describes the behavior of Web Services.

- **Goals:** describe aspects related to user desires with respect to the requested functionality, i.e. they specify the objectives of a client when consulting a Web service. Thus they are an individual top-level entity in WSMO.

- **Mediators:** describe elements that handle interoperability problems between different elements, for example two different ontologies or services. Mediators can be used to resolve incompatibilities appearing between different terminologies (data level), to communicate between services (protocol level), and to combine Web services and goals (process level).

Besides these main elements, **Non-Functional** properties such as accuracy, network-related QoS, performance, scalability, and reliability are used in the definition of WSMO elements that can be used by all its modeling elements. Furthermore, there is a formal language to describe ontologies and Semantic Web services called WSMX (Web Service Modeling Language) which contain all aspects of Web service descriptions identified by WSMO. In addition, WSMX (Web Service Modeling eXecution environment) is the reference implementation of WSMO, which is an execution environment for business application integration [16].

WSMO group [9] refers to the problem of WSC as orchestration. A graphical tool in [17] is presented to guide the user to compose a process, but no additional computational support or automation is present.

### 3. MOTIVATION

#### 3.1 Motivating Scenario

The motivating example that we illustrate in this section is based loosely on a scenario described in [22]. Suppose Hassan wants to participate on one international conference in Kuala Lumpur for a given period of time (e.g., staying there from December 1 to December 5). Therefore, it is not sufficient to register, but he should also take care of submitting the camera-ready version of the paper, booking a flight, reserving a hotel, renting a car, and weather forecast.

For the above activities, there are the following preferences and constraints:

- The starting date is specified after getting information regarding weather forecast.
- The paper submission is allowed only after the paper is registered, i.e. after registration fee payment is made.
- The participant can pay all fees such as ticket price, hotel rate, car rental and registration fee with credit card at once.
- The hotel has to be booked according to the flight date (i.e., if the flight arrives on December 1, then the hotel has to be booked from December 1).

In order to fulfill the above request, which is depicted in Figure 1 and illustrated with WSML notation [17], we need an automated Online Conference Registration System (OCRS) which can find an execution path based on these predefined task decompositions, and then we can perform Hassan’s Web Service discovery and composition task automatically. Indeed, the participant sends his request to the OCRS, which shows the weather forecast based on the request, and then the OCRS has to build a package including a travel to/from Kuala Lumpur, a hotel for all the nights spent in Kuala Lumpur, and a car based on the dates which are specified by the participant. Finally, the paper submission system has to be offered by the OCRS. As an example, the Web service BookFlight which can be seen in Figure 2 is one of the candidate Web services for the OCRS to satisfy the goal.
3.2 Objectives

The objectives of this research are as follows:

- To evaluate state of the art of Web service composition approaches.
- To investigate and develop a novel approach called AIMO for automated Web service discovery and composition based on AI-planning and Semantic Web for minimizing the time and cost of Web service composition.
- To develop a composer tool that supports the proposed approach.
- To compare the proposed approach based on some criteria with other current approaches.

It is important to note that this research doesn’t want to give any preference to cost over time. Thus, the service with the smallest cost will not necessarily be part of composition, since its other measures of quality must be considered.

4. METHODOLOGY

In [34], the authors propose, as hypothesis, a classification of the research problems of the software engineering discipline in: Engineering problems, concerned with the construction of new objects such as methodologies, approaches, techniques, etc; Scientific problems, concerned with the study of the existing object such as algorithmic complexity, software metrics, testing techniques, etc. In the case of Engineering, it would be necessary to study existing methodologies, reflecting on them to determine their advantages and disadvantages and proposing a new one, which, while retaining the advantages of the methodologies studied, would, as far as possible, lack their shortcomings.

This research has aimed to develop a novel approach for discovery and composition of Semantic Web services. Thus, producing this approach is an engineering problem and this research will focus on engineering design such as modeling, constructing, and evaluating the new object.

This research can be defined in the scope that contributes to the Semantic Web services, Web service discovery, Web service composition and Web Intelligence. Some existing algorithms or approaches need to be revised and enhanced in order to achieve the objectives of the study. In Figure 3, the flow chart of the research activities has been shown.

Figure 1. The user’s Goal

![Figure 1](http://example.org/OnlineConferenceRegistrationSystem)

Figure 2. Capability Level of the Flight Booking Service

![Figure 2](http://example.org/BookFlight)

Figure 3. Research Flow Chart

![Figure 3](http://example.org/)
5. RESULTS

5.1 Formalizing WSD and WSC Problems

Based on Description Logics, the Web service discovery algorithm, which can be seen in Figure 4, is proposed and defined as follows:

Definition 1 (Web Service Discovery) Suppose that a requested capability of a Goal $G_1$ is given. Let a capability of a Web service $C_W$ be discovered. The Web Service Discovery is defined as to find a set of Web services $W$, such that: $(C_W \sqsubseteq C_{W_1}) \lor (C_W \sqsubseteq C_{W_2}) \lor ... \lor (C_W \sqsubseteq C_{W_k}) \lor \neg (C_W \sqsubseteq C_{W_1} \sqsubseteq ... \sqsubseteq C_{W_k})$.

![Algorithm 1 WSDiscovery (Guser, Gexist, W)](image)

Figure 4. Web service discovery algorithm

The workflow-based approaches are usually used in the situation where the request has already defined the process model, but automatic program is required to find the atomic services to complete the requirement. The AI-planning based approaches deal with Web services as planning operators and use a causal planner to generate WSC. The syntactic-based approaches concentrate on two main approaches, namely: orchestration and choreography. Choreography languages are still in an introductory phase of definition. In Semantic-based approaches, ontologies are used as data models throughout these types of approaches, meaning that all resource descriptions and all data exchanged during service usage are based on ontologies. The main problems with most of these approaches to WSC are the verification of correctness of WSC and the analysis of QoS aspects. (More details on WSC approaches can be found in [33]).

5.3 AIMO Architecture

In order to realize the proposed approach, some specific extension has to be added to the WSMO framework. Figure 5, shows main components of the proposed framework for the Web service composition. The architecture consists of four main components:

- **EHTN-DL Planner:** This component is the heart of the framework, where composition of services for making higher level ones is done. The component is responsible to compose two or more operators or methods to satisfy the task request using task decomposition. The planner uses the WSML-DL reasoner to find the matching operators and methods for a given task. The reasoner is again used by the planner to evaluate the preconditions of actions to determine applicability. Note that in this framework, if an atomic Web service can satisfy the user’s goal, the planner will not be used.

- **AIMO Translator:** This component is used in the approach to translate each WSMO capability of service into an element in the task ontology (Tont) using WSMO ontologies. In addition, the translator provides translating of WSMO service interfaces into a set of methods and operators.

- **WSMX:** This component has a component-based architecture and is a reference implementation of WSMO. In addition, it takes the full conceptual model of WSMO into consideration. Some of its components are used in this framework such as discovery, data mediation, and process mediation. Furthermore, the architecture takes care of non-functional properties during matching the semantic capability descriptions of Web services and goals, using discovery component of WSMX.

- **WSML-DL Reasoner:** This component captures the expressive Description Logic SHIQ(D) and consists of a wrapper of WSML-DL expressions to a classical Description Logics syntax [18]. It is possible to perform, among others, the reasoning tasks of checking ontology consistency, entailment and instance retrieval, using this component. According to [15], WSML-DL is semantically equivalent to OWL DL. However, WSML uses an epistemology which abstracts from the underlying logical language, whereas OWL directly uses description logic’s epistemology.

In addition, the definition of Web service composition based on description Logic is proposed as follows:

Definition 2 (Web Service Composition) Suppose that a requested Goal $G_1$ is given. Let a set of Web services which are discovered $W$. The Web Service Composition is defined as a five-tuple $<S; G_o; G_{exist}; W; D>$ to select a finite parallel or sequence set of Web services $(W_1, ..., W_k)$ such that: $(G_o \sqsubseteq (W_1 \sqcup ... \sqcup W_k))$.

- $S$ is the initial state.
- $G_{exist}$ is a set of existing goals as defined in WSMO goals.
- $D$ is the planning domain as defined in HTN-DL.

5.2 Evaluation of Current WSC Approaches

In this section, WSC approaches are categorized into four categories. Then, these approaches are compared based on some criteria (like QoS, scalability, and correctness). The WSC approaches can be classified using the following four aspects:

1) Workflow based approaches like EFLOW [24] and Polymorphic Process Model (PPM) [25].
2) AI-Planning based approaches like HTN-DL, which is described before and Situation Calculus [19].
3) Syntactic based approaches like BPEL4WS [20] and WS-CDL [30].
4) Semantic based approaches like WSMO [9], which is described in before and OWL-S [32]. Notice that, the boundaries between these four aspects of classification are not always strictly defined. BPEL4WS, for example, can be also considered as a workflow-based approach.
In the following, the mechanism for using WSMO and the planner is described. Firstly, the users’ request is presented to the WSMO with a WSMO Goal that formally describes what they would like to achieve.

In the next step, WSMX uses the Discovery component to find Web services, which have semantic descriptions registered with WSMX that can fulfill this Goal. Goal is represented through the capability and interface of the Web services the user would like to have, and interact with. A set of properties strictly belonging to a goal are defined as non-functional properties of a WSMO goal. A goal may be defined by reusing one or several already-existing goals by means of goal mediators.

During the discovery process the users Goal and the Web services description may use different ontologies. If this occurs Data Mediation is needed to resolve heterogeneity issues. Data Mediation in WSMX is a semi-automatic process that requires a domain expert to create mappings between two ontologies that have an overlap in the domain that they describe. Once these mappings have been registered with WSMX, the runtime data Mediation component can perform automatic mediation between the two ontologies. Once this mediation has occurred and a given service that can fulfill the users Goal has been chosen, WSMX can begin the process of invoking the service.

Every Semantic Web service has a specific choreography that describes the way in which the user should interact with it. This choreography describes semantically the control and data flow of messages the Web Service can exchange. In cases where the choreography of the user and the choreography of the Web Service do not match, process mediation is required. The Process Mediation component in WSMX is responsible for resolving mismatches between the choreographies of the user and Web service.

Based on the motivating scenario, we give an example of discovery services for flight tickets whose goals can be satisfied using WSMO Web services. Therefore, the AIMO doesn’t need to use EHTN-DL planner. The goals are as follows:

1) Flight tickets from all places in France to all places in Malaysia ($G_1$).

2) Flight tickets from Paris to Kuala Lumpur ($G_2$).

Note that to make the example as simple as possible, terminological mismatches are not included.

Then we use our algorithm 1 to search all of the available services and get the appropriate candidate services to satisfy the Goals. The result ($w_1$) and goals can be seen in Figure 6.

$$G_1 : \forall p_2, p_3 ((p_2, \text{France}) \land (p_3, \text{Malaysia}) \rightarrow 3p_2, \text{flightTicket}(p_1, p_2, p_3))$$

$$G_2 : 3p_1, \text{flightTicket}(p_1, \text{Paris}, \text{KualaLumpur})$$

$$W_1 : \forall p_2, p_3 ((p_2, \text{Europe}) \land (p_3, \text{Asia}) \rightarrow 3p_2, \text{flightTicket}(p_1, p_2, p_3))$$

**Figure 6. Examples of goals and a service that matches to them**

If no single Web service can satisfy the request then the request is offered to the planner. The planner then tries to combine existing Semantic Web services and generate the process model. In the proposed framework, the process generator is based on AI-planning. The process generator to tackle the problems of heterogeneous ontologies and choreography uses discovery component of WSMX. Thus via this component, the process generator will be able to discover the appropriate Semantic Web services for the composite process. Finally the process model will be offered to the WSMX Invoker for execution. The stages for execution of Web services as a process model are like as single Web services that were already described. (More details on AIMO architecture and planning process can be found in [29]).

### 5.4 Implementation

In order to validate the feasibility of the proposed approach, we have developed a prototype **AIMO-Composer**. The prototype is based on Java, open-source components, and the NetBeans development environment. So far, the AIMO-Composer provides the following functionalities: 1) Translating the Web service descriptions from WSDL into WSMO in order to be adapted with the discovery component of WSMO, which is described in section 5.3. 2) Discovering the Web Services through the way of manipulating the relations between ontologies and based on the...
functionalities and some non-functional properties that are specified by the user as a goal at process run-time. 3) Transformation of the discovered Web services to EHTN-DL. 4) Composing the discovered Web services based on the proposed algorithm.

We started with an implementation of AIMO translator which is a Java program that reads in a collection of WSMO process definitions and outputs a EHTN-DL domain.

The discovery component, which is based on WSMO, provides searching capabilities for discovering Web services that satisfy certain requirements. Such requirements could be based on the currently available set of input data and the expected output data. Moreover, semantic Web service concepts are implemented here to enable semantic search and discovery of suitable Web services.

The composer component is responsible for building complex services. Apart from applying the EHTN-DL planner other planning and optimization software may be used.

Therefore, as depicted in Figure 5, these components publish their functionality via SOAP interfaces and can be utilized in the AIMO framework as infrastructure services in order to discover services or perform compositions.

Once all the input parameters are provided, AIMO starts the planning process using the domain description obtained from the translation of the WSMO files. EHTN-DL decomposes the top level task into smaller subtasks, and absolutely there may be multiple different decompositions for any given task.

The output of AIMO-Composer is a sequence of world-altering Web Services calls that can be subsequently executed.

5.5 Experimental Validation

In order to evaluate Web service composition process the following criteria can be considered: execution duration, execution price, reputation, reliability and availability of Web services. In this paper, we concentrate on execution duration of the planning process which is measured as delay between the moment when a request is sent and the moment when the results are received.

We have used the Semantic Web Services Test Case version 1.1 (accessible at http://projects.semwebcentral.org/frs/download.php/277/SWS-TC-1.1.zip) as our Web services repository that includes 241 Web services.

For evaluation of the proposed approach using the repository above, we have used the motivating scenario described in section 3.1 and the result can be seen in Table 1.

<table>
<thead>
<tr>
<th>Number of Web services</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>241</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Duration (s)</td>
<td>15.6</td>
<td>27.2</td>
<td>45.8</td>
<td>62.2</td>
<td>72.3</td>
</tr>
</tbody>
</table>

As the result of the experiment shows, the proposed approach found all plans for the case study in a timely manner.

6. DISCUSSION

The closest related works to the proposed approach are HTN-DL [10] and WSMO [9] which already described. Other current approaches which are being used to address the issue of WSC and similar to the proposed framework are HTN-planning based approaches described in [7] and [13]. Sirin et al. [13] proposed an approach to translate process models and service descriptions written in OWL-S into the problem and domain description language of the HTN planner SHOP2 [21], and then, shows how to solve service-composition problems using SHOP2 based on that translation methodology.

Other approaches based on planning, such as planning as model checking, are being considered for Web service composition and would allow more complex constructs such as loops [7]. Some researchers propose AI-planning based WSC approaches, such as using situation calculus [19] and GOLOG [4]. These approaches can be useful for discovery, composition, and contracting of Web services.

Rao et al. [5] focus on the discovery and composition of Web services and propose a mixed initiative framework for semantic Web service discovery and composition. They discuss the use of the GraphPlan algorithm to successfully generate a process. One weakness of their work is the lack of mediators both in data and process levels.

The Business Process Execution Language for Web Services (BPEL4WS) [20] addresses compositions where the control flow of the process and the bindings between services are known in advance.

Based on our evaluation of the state-of-the-art approaches for WSC [33], the HTN-DL formalism can be considered as an optimized AI-planning technique. That is because of the following reasons:

1) The hierarchical structure of HTN-DL domains can conveniently describe composite Web Service descriptions and fit in well with the loosely coupled nature of Web Services. Ontology-based reasoning provides flexible mechanism to reuse the Web Services that are defined by separate developers in different contexts.

2) The components of the planning system, the OWL-DL reasoner Pellet and the API for OWL-S services are also released as standalone tools and have been incorporated in many systems.

However this formalism still has some problem to be solved, which can be addressed as follows:

1) An important problem HTN-DL has not addressed is the process of discovering Web Service descriptions. The HTN-DL planner assumed that all the service descriptions would be provided as input to the system. This is not very realistic as we cannot expect the user’s planning agent to crawl the Web to find all available Web services. However, we tackle this problem by using discovery element of WSMO.

2) The planning system provided by HTN-DL is based on OWL-S. However this technique itself is problematic. For example Non-functional properties in OWL-S are restricted to the service profile. However, this can be expressed in all elements of WSMO that we use it in our approach.
7. CONCLUSION AND FUTURE WORK
In this paper, we have summarized ongoing work on the development of a Semantic Web service discovery and composition approach, called AIMO. In contrast to most work in this field, our approach concerns the use of both Web service discovery and composition.

Key contributions of this paper include: provision of the AIMO framework that can compensate lack of both discovery in HTN-DL formalism and appropriate composition process in WSMO (i.e., orchestration), provision of a translator for interaction between WSMO and AI-planning, and provision of the hybrid algorithm of HTN-DL and WSMO, called EHTN-DL which can generate a concrete planning process and realize service composition automatically. This approach enhances the performance of HTN-DL and correctness verifiability of WSMO. Moreover, we have implemented some parts of the AIMO architecture and based on this implementation, and experiments, the proposed approach is useable and performed.

We are currently planning to generalize our approach described in this paper to do planning for another complex case study namely Online Banking System (OBS).

8. REFERENCES
[16] WSMX working group. DOI= http://www.wsmx.org