Security conscious AI-planning-based composition of semantic web services

Sayed Gholam Hassan Tabatabaei
Universiti Teknologi Malaysia (UTM), Kuala Lumpur, Malaysia
Amir Vahid Dastjerdi
The University of Melbourne, Melbourne, Australia
Wan M.N. Wan Kadir and Suhaimi Ibrahim
Universiti Teknologi Malaysia (UTM), Kuala Lumpur, Malaysia, and
Elahe Sarafian
Shahid Bahonar University of Kerman, Kerman, Iran

Received 16 December 2009
Revised 21 March 2010
Accepted 13 May 2010

Abstract

Purpose – Automated composition of semantic web services has become one of the recent critical issues in today's web environment. Despite the importance of artificial intelligence (AI)-planning techniques for web service composition, previous works in that area do not address security issues, which is the focus of this paper. The purpose of this paper is to propose an approach to achieve security conscious composition of semantic web services.

Design/methodology/approach – The proposed approach called security conscious composition of semantic web services (SCAIMO) is based on the prior work, i.e. AIMO. The AIMO is an effective approach for web service discovery and composition based on AI-planning, web service modeling ontology (WSMO), and description logic (DL). In this paper, definitions of secure matchmaking and web service composition are formalized based on DLs. Moreover, security capabilities and constraint types in the proposed SCAIMO framework are presented.

Findings – This paper proposes a secure task matchmaker which is responsible for matching security conscious tasks with operators and methods based on WSMO and DL to support the proposed SCAIMO framework. In addition, the paper implements and evaluates the SCAIMO using a test case and the result shows that the approach can provide an applicable solution.

Originality/value – The key contribution of this paper encompasses the new framework to support security capabilities and constraints during composition of semantic web services as well as the new secure task matchmaker.

Keywords World wide web, Data security, Semantics

Paper type Research paper

1. Introduction

Nowadays, the term "web services" has been used very often. According to W3C (Web Services Architecture Requirements, 2004):

This research is supported by the Ministry of Science and Technology and Innovation, Malaysia and Universiti Teknologi Malaysia under the vote No. 79364.

International Journal of Web Information Systems
Vol. 6 No. 3, 2010
pp. 203-229
© Emerald Group Publishing Limited
1744-0084
DOI 10.1108/17440081011070150

A Web service is a software system identified by a Uniform Resource Identifiers (URI) (Berners-Lee *et al.*, 1998), whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

In today's web environment, a number of those companies which are planning to implement the core of their business as a service on the web are growing dramatically. 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), compose (automatically), discover, secure, and ensure the correctness of web services. In this paper, a significant portion of our work has been dedicated to scenarios aimed at secure automating web service composition functionality.

In some cases, if a single web service does not fulfill the service requestor's goals, the current web services should be combined together to achieve the goal. That issue has inaugurated a number of studies on the area of web services composition (WSC) both in industry and academia.

Most of the current approaches related to WSC applied following techniques: hierarchical task network (HTN) (Sirin *et al.*, 2005), Golog (McIlraith and Son, 2002), classic artificial intelligence (AI)-planning (Rao *et al.*, 2006), rule-based planning (Medjahed *et al.*, 2003), model checking (Kuter *et al.*, 2005), theorem proving (Rao *et al.*, 2004), etc. However, considering security issues in WSC have not been carefully addressed by most of the previous works. In this paper, we propose an automated approach for *S*ecurity *C*onscious composition of semantic web services, called SCAIMO, in order to tackle that problem. The aim is to insert security consideration for composition into our prior work, i.e. *AIMO* (Tabatabaei *et al.*, 2008a, b) which is an effective approach for WSC based on *AI*-planning, web service modeling ontology (WS*MO*) (WSMO Spec., 2005), and description logic (DL) (Baader *et al.*, 2003).

What needs to be considered in security of WSC can be classified as satisfying service provider and requestor security requirements or constraints. The first step is to describe those constraints and security features for web services. In this work, WSMO has been applied to add security constraints and capabilities for web service descriptions and queries. We further show how composition can be done by considering security requirement of providers and requestors. In addition, an implemented component called secure task matchmaker is proposed to be added to the AIMO architecture in order to perform requirement-capability matchmaking.

The remainder of this paper is organized as follows. We first give some background information of related ideas in Section 2. We then motivate our work through a proper scenario in Section 3. We formalize definitions of secure matchmaking and WSC based on DLs in Section 4. A comparative evaluation of state-of-the-art approaches for WSC, including security aspects, is presented in Section 5. The next two sections form the technical core of this work, Section 6 describes security capability and constraint types in SCAIMO, and Section 7 details the proposed SCAIMO architecture. The results of implementation and evaluation of the SCAIMO approach are presented in Sections 8 and 9, respectively. Finally, the paper is concluded in Section 10.

2. Preliminaries

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

2.1 HTN planning and HTN-DL formalism

In order to perform tasks or actions, a planner called HTN is applied to generate a sequence of activities which can satisfy those tasks or actions. On the other hand, the key attribute of web services (which is loosely coupled) is adopted with domains of the HTN. That is because of autonomous feature of task decompositions such that the methods' designer does not need to have close information about how previous or next decompositions are happened. The HTN applies a forward-decomposition for the planning process. In that case, each planning problem is identified as a tuple (S, w, D) such that the initial state is denoted by S, the task network is denoted by w, and the planning domain is denoted by D. The planning domain is described by a collection of methods and operators as well as an ontology of DL namely task ontology (T_{ont}). Therefore, concept and individual names of the task ontology are corresponding with task symbols and method names, respectively. The world state is kept by the HTN planning and actions are applied by the HTN for representing state transitions like classical planning. However, what and how the classical and the HTN planners plan for it, are different from each other.

The mechanism of the HTN planner, which is a forward-searching algorithm, is as follows: first of all, the task with no predecessors is selected by the planner. If the task is primitive, the planner tries to find an appropriate operator and add it to the plan. Otherwise, the planner for the non-primitive tasks tries to find an appropriate method such that the chosen task will be substituted with the subtasks of that method. The process of task decompositions is continued while no task can be found in the network. More information about HTN planning can be seen in Ghallab *et al.* (2004).

Though many obstacles for web service composition are solved by means of the HTN planning (Sirin *et al.*, 2004), yet there are few limitations which can be seen for direct usage of HTN planning in web services as follows (Sirin, 2006):

- In order to understand relations between various tasks in HTN planning, no way is provided to make inference. In that case, the name and the number of parameters of each task are specified.
- Owing to some limitations on non-primitive and primitive tasks, it is not possible to have both composite and atomic services satisfying the same task or request.
- Owing to the limitation of pear-to-pear mapping between primitive tasks and operators, a specified task can be satisfied by only one operator. However, since there might be possibility of satisfying a given primitive task by various web services, the above issue is not true for web services.

The HTN-DL approach, which is proposed by Sirin (2006), tries to tackle the web service composition problem automatically by combining HTN-planning and DLs (Baader *et al.*, 2003). The HTN-DL is demonstrated by Web Ontology Language (OWL).

Moreover, the DL (Baader *et al.*, 2003) is applied by HTN-DL for description of states and actions using an expressive language for knowledge representation. In that case, a DL knowledge base represents the state of the world. In addition, in order to have flexible matchmaking, the task ontology is responsible to provide descriptions of non-functional properties as well as categorization of web services. Furthermore, during the composition process, the knowledge and world-altering effects are differentiated by the HTN-DL. Using that way, it is possible to provide interleaving planning with invocation in order to invoke information-providing services.

In addition, a set of translation algorithms to provide mapping from OWL-S Spec. (2004) into HTN-DL is proposed by the author. Using that translator, it might be possible to include semantics into the OWL-S process model.

The strength point of HTN-DL is providing efficient reasoning based on DLs for the HTN planning. In fact, the planning system is directly related to those service interfaces which are performed through the DL reasoner. However, there are still some limitations in HTN-DL that this research tries to address them which are listed in Section 5.3.

2.2 Description logics

DLs (Baader *et al.*, 2003) are a family of knowledge representation formalisms that are able to represent the structural knowledge of an application domain through a KB 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) *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; and

(3) inferences which are reasoning mechanisms of Tbox and Abox.

In the rest of the section, we briefly describe the syntax and semantics of the DL $\mathcal{SHIQ}$ (Horrocks *et al.*, 2000; Hustadt *et al.*, 2004) which is the DL underlying web service modeling language (WSML-DL) (Steinmetz, 2007).

Let N_R be the set of role names. The set of $\mathcal{SHIQ}$ roles is the set $N_R \cup \{R^- | R \in N_R\}$. For $R \in N_R$, let $\mathsf{Inv}(R)$ denote R^- and let $\mathsf{Inv}(R^-)$ denote R. An RBox $\mathcal{R}$ over N_R is a finite set of transitivity axioms $\mathsf{Trans}(\mathrm{R})$ and role inclusion axioms $R \sqsubseteq S$, where R and S are roles, such that, if $R \sqsubseteq S \in \mathcal{R}$, then $\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \in \mathcal{R}$ as well. Let $\sqsubseteq^*$ denote the reflexive-transitive closure of $\sqsubseteq$. A role R is transitive if $\mathsf{Trans}(S) \in \mathcal{R}$ or $\mathsf{Trans}(\mathsf{Inv}(S)) \in \mathcal{R}$ for some S with $S \sqsubseteq {}^*R$ and $R \sqsubseteq {}^*S$; R is simple if there is no role S such that $S \sqsubseteq {}^*R$ and S is transitive; R is complex if it is not simple.

Let N_C be a set of atomic concept names. The set of $\mathcal{SHIQ}$ concepts over N_C and N_R is defined inductively as the smallest set for which the following holds: $\top$ and $\bot$ are $\mathcal{SHIQ}$ concepts, each atomic concept name $A \in N_\mathrm{C}$ is a $\mathcal{SHIQ}$ concept, if C and D are $\mathcal{SHIQ}$ concepts and R is a role, then: $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$ are also $\mathcal{SHIQ}$ concepts, and, if C is a $\mathcal{SHIQ}$ concept, R a simple role and n an integer, then $\leq nR.C$ and $\geq nR.C$ are $\mathcal{SHIQ}$ concepts.

For example, if we suppose that *Person* is an atomic concept and *hasChild* is an atomic role, using the bottom concept ($\bot$), we can form the concept $Person \sqcap \forall hasChild.\bot$, denoting those persons without a child.

TBox $\mathcal{T}$ over N_C and $\mathcal{R}$ is a finite set of concept inclusion axioms $C \sqsubseteq D$ or concept equivalence axioms $C \equiv D$, where C and D are $\mathcal{SHIQ}$ concepts.

Let N_I be a set of individual names. An ABox $\mathcal{A}$ 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 $\mathcal{SHIQ}$ concept, R a role, and a and b are individuals.

A $\mathcal{SHIQ}$ *KB* is a triple of the form $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$, where $KB_{\mathcal{R}}$ is an RBox, $KB_{\mathcal{T}}$ is a TBox, and $KB_{\mathcal{A}}$ is an ABox.

The semantics of a $\mathcal{SHIQ}$ *KB* is given by the mapping π which transforms *KB* axioms into a set of first-order formulae. *KB* is satisfiable if $\pi(KB)$ is satisfiable.

Regarding a *KB* $\mathcal{T}$, $\mathcal{A}$, several types of reasoning problems are supported by a DL reasoning system. In this paper, we just introduce the concept of satisfiability and subsumption:

- *Concept satisfiability*. A concept *C* is satisfiable w.r.t. *KB* if there exists a model of *KB* in which the interpretation of *C* is not empty. This is the case iff $KB \cup C(\alpha)$ is satisfiable, where α denotes a new individual not occurring in *KB*.
- *Subsumption*. A concept *C* is subsumed by a concept *D* w.r.t. *KB* if $\pi(KB)| = \pi(C \sqsubseteq D)$. This is the case iff $KB \cup (C \sqcap D)(\alpha)$ is unsatisfiable.

2.3 Web service modeling ontology

Semantic web services can be described by a model called WSMO. The conceptual basis of this model is set up in the Web Service Modeling Framework (WSMF) (Fensel and Bussler, 2002). WSMO defines four key concepts as follows (Fensel *et al.*, 2006): goals, web services, ontologies, and mediators. In the following, descriptions of these concepts are presented.

2.3.1 Ontologies. In order to develop semantic web services successfully, this element is considered as the key concept by providing the (domain specific) terminology. In addition, to provide a link between human terminologies and machine, the formal semantics can be applied by ontologies. The concepts and relations of ontologies (i.e. their epistemologies) are described by WSMO.

2.3.2 Web services. In order to provide domain specific values, this concept as a computational entity might be applied. The key elements for description of a service are stated as follows:

- capability is responsible to define the value provided by each service; and
- interfaces which can be one or more to describe the service choreography and the orchestration.

2.3.3 Goals. The goals are responsible to identify the service requestor's requirements based on her desired capabilities. In fact, the user's purposes once contacting with a web service are specified by goals, which are one of the key independent elements in WSMO. On the other hand, goals can be defined as descriptions of web services which can fulfill the service requestor's requirements.

2.3.4 Mediators. The problems of interoperability between WSMO elements can be handled by a WSMO element called mediator. WSMO defines three mediation levels with respect to different semantic web services requirements for mediation which are described as follows:

(1) data-level mediation to solve the heterogeneity problems between transfer protocols and data sources;

(2) functional-level mediation to tackle the heterogeneity problems between capabilities of goals and web services; and

(3) process-level mediation to tackle the heterogeneity problems between business processes and communication protocols.

Apart from the above key elements, WSMO defines non-functional properties which can be applied by all other WSMO concepts. In addition, WSMO identifies a formal language namely WSML for describing semantic web services and ontologies. The WSML contains features of description of services defined by WSMO. Moreover, as a reference development of WSMO, an execution environment called Web Service Modeling eXecution environment (WSMX) is identified by the WSMO group to integrate business applications (WSMX Spec., 2005). Therefore, using such elements, it is possible to improve automation of business processes along with improving scalability.

WSMO Spec. (2005) refers to the problem of WSC as orchestration. A graphical tool is presented to guide the user to compose a process, but no additional computational support or automation is present.

3. Motivating scenario

Suppose Hassan wants to participate on one international conference in Kuala Lumpur for a specified period of time (e.g. 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 constraints and preferences:

- 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.
- In order to book the hotel, the flight date has to be taken into account such that the participant arrives on December 1, then the date of booking hotel has to be started from December 1.

Moreover, following constraints have to be taken into consideration from security perspective:

- AES encryption is required for all web services.
- Hassan is only capable of supporting X.509 authentication mechanism and AES encryption.

In order to fulfill the Hassan's request, which is shown in Figure 1 and illustrated with WSML notation (WSML Spec., 2005), we need an automated Online Conference

```
goal _"http://example.org/OnlineConferenceRegistrationSystem"
 capability _"http://example.org/OnlineConferenceRegistrationSystem#capl"
 sharedVariables {?date, ?origin, ?participant, ?authentication, ?encryption}
 precondition definedBy
              ?date memberOf _date and
              ?origin memberOf conf#Location and
              ?participant memberOf conf#participant and
              ?authentication memberOf scr#authentication and
              ?encryption memberOf scr#encryption
 assumption definedBy exists
 ...
 effect definedBy exists
 ...
```

Figure 1.
Security-related goal constraints

Registration System (OCRS). OCRS 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.

Apart from security constraints which are described in Section 6, the web service *BookFlight* is one of the candidate web services for the OCRS to satisfy the goal. The *BookFlight* service can be shown in Figure 2.

The remainder of the article focuses on matching process between the request and the service including security constraints, as well as the case where no single web service can fulfill the request and, therefore, one has to compose multiple web services. These types of such problems are referred to as secure matchmaking and WSC problems, respectively, which are elaborated in subsequent section.

4. Formalizing the secure matchmaking and the WSC problems

Before we proceed to define secure matchmaking and WSC, we need to introduce the security conscious goal and five matching operations described below.

Definition 1 (security conscious goal). Let $\mathcal{T}$ be an acyclic Tbox. A security conscious goal G for $\mathcal{T}$ is defined in the form of $G = (C_G, I_G, N_G)$, where:

- C_G is the set of capabilities of web services including security-related goal constraints, which the user would like to have.

```
webService _"http://example.org/BookFlight"
capability _"http://example.org/BookFlight#cap1"
 sharedVariables {?date, ?origin, ?dest, ?client}
 precondition definedBy
     exists {?req}
             (?req[
                trv#date hasValue ?date,
                trv#origin hasValue ?origin,
                trv#destination hasValue ?dest,
                fl#client hasValue ?client] memberOf fl#FlightRequest).
 assumption definedBy
     exists {?flight}
             (?flight [
                trv#origin hasValue ?origin,
                trv#destination hasValue ?dest,
                trv#date hasValue ?date] memberOf fl#FlightAvailable).
 postcondition definedBy
             _#offer [trv#date hasValue ?date,
                     trv#origin hasValue ?origin,
                     trv#destination hasValue ?dest,
                     trv#flightNumber hasValue #fn,
                     fl#client hasValue ?client] memberOf fl#FlightOffer.
 effect definedBy
            _#booking[ trv#date hasValue ?date,
                      trv#origin hasValue ?origin,
                      trv#destination hasValue ?dest,
                      fl#flightNumber hasValue #fn,
                      trv#pax hasValue ?client] memberOf fl#FlightBooked.
```

Figure 2.
Capability level of the BookFlight service

- I_G is the set of interfaces of web service, which the user would like to have and interact with.
- N_G is the set of non-functional properties, which is similar to that attached to web services.

The above definition is illustrated in more detail in Section 6.

Definition 2 (exact matching). Suppose that a requested capability of a security conscious goal $C_{G1} \in G_1$ is given. Let a capability of a web service $C_{W1} \in W$. If $C_{G1} \equiv C_{W1}$ then G_1 can "exactly" match W_1, i.e. $G_1 \equiv W_1$.

Definition 3 (PlugIn matching). Suppose that a requested capability of a security conscious goal $C_{G1} \in G_1$ is given. Let a capability of a web service $C_{W1} \in W_1$. If $C_{G1} \sqsubseteq C_{W1}$ then $G_1 \sqsubseteq W_1$. This match is called "PlugIn".

Definition 4 (subsumption matching). Suppose that a requested capability of a security conscious goal $C_{G1} \in G_1$ is given. Let a capability of a web service $C_{W1} \in W_1$. If $C_{W1} \sqsubseteq C_{G1}$ then $W_1 \sqsubseteq G_1$. This match is called "subsumption".

Definition 5 (intersection matching). Suppose that a requested capability of a security conscious goal $C_{G1} \in G_1$ is given. Let a capability of a web service $C_{W1} \in W_1$. If $\neg (C_{G1} \sqcap C_{W1} \sqsubseteq \perp)$ then $\neg (G_1 \sqcap W_1 \sqsubseteq \perp)$. This match is called "intersection".

Definition 6 (non-matching). Suppose that a requested capability of a security conscious goal $C_{G1} \in G_1$ is given. Let a capability of a web service $C_{W1} \in W_1$. If $C_{G1} \sqcap C_{W1} \sqsubseteq \perp$ then $G_1 \sqcap W_1 \sqsubseteq \perp$. This relationship is called "non-match".

Based on the above definitions, we propose the definition of secure matchmaking as follows.

Definition 7 (secure matchmaking). Suppose that a requested capability of a security conscious goal $C_{G1} \in G_1$ is given. Let a capability of a web service $C_{Wi} \in W_i$. Secure matchmaking is defined as to find a set of web services W_i such that:

$$(C_{G1} \equiv C_{W1}) \sqcup (C_{G1} \sqsubseteq C_{W1}) \sqcup (C_{W1} \sqsubseteq C_{G1}) \sqcup (\neg (C_{G1} \sqcap C_{W1} \sqsubseteq \perp)).$$

We propose the formal definition of web service composition problem as follows.

Definition 8 (secure web service composition). Suppose that a requested security conscious goal G_1 is given. Let a set of web services which are discovered W. The secure web service composition is defined as a five-tuple $< S, G_1, G_{exist}, W, D >$ to select a finite sequence set of web services $(W_1, \ldots, W_k)$, such that: $G_1 \sqsubseteq (W_1 \sqcup, \ldots, \sqcup W_\mathrm{k})$, where:

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

In the motivating scenario, the type of matching between the security conscious goal, i.e. Hassan's request (G_1), and *BookFligh* web service (W_1) is subsumption because $C_{W1} \sqsubseteq \mathrm{C}_{G1}$ therefore $W_1 \sqsubseteq G_1$. The same matching can be seen between the goal and other web services. Therefore, set of discovered web services $(W_1, \ldots, W_k)$ have to be combined based on Definition 8 such that $G_1 \sqsubseteq (W_1 \sqcup, \ldots, \sqcup W_k)$. To address this intractable WSC problem, we will propose an AI-planning based approach called SCAIMO in the following sections.

5. Comparative evaluation of state-of-the-art WSC approaches

In this section, we categorize WSC approaches and compare these approaches based on some criteria, like security, quality of service (QoS), scalability, and correctness. More details can be found in Tabatabaei *et al.* (2008a, b).

5.1 Classification

We can classify the WSC approaches using the following four aspects. Notice that, the boundaries between these four aspects of classification are not always strictly defined. For example, Business Process Execution Language for Web Services (BPEL4WS) (BPEL4WS Spec., 2007) as a syntactic-based approach can be also categorized as a workflow-based approach.

5.1.1 Workflow-based approaches. Workflow-based composition methods can be distinguished from the static and dynamic workflow generation. The static composition means that the requester before starting the composition planning should build an abstract process model. However, in dynamic composition, creating process model and selecting single web services are done automatically. The two major approaches based on workflow are EFlow (Casati *et al.*, 2000) and polymorphic process model (PPM) (Shuster *et al.*, 2000).

5.1.2 AI-planning-based approaches. Currently, several approaches based on AI-planning have been presented to solve the problem of WSC. Most of these approaches rely on the model of state-transition system. In this system, there are finite or recursively countable set of states, actions and events along with a transition function that maps a state, action, event tuple to a set of states. The goal of planning is to find which actions to apply to which states in order to achieve some objective, starting from some given situation. Basically, classical planning is based on the initial modeling of the STRIPS (Fikes and Nilsson, 1990) system. The two principal approaches based on AI-planning are HTN-DL, which is described in Section 2.1 and situation calculus (Hakimpour *et al.*, 2005), which is a first-order language for reasoning about action and change.

5.1.3 Syntactic-based approaches. Currently, there are two main approaches in the field of syntactic-based WSC:

(1) Web service orchestration combines available web services by adding a central coordinator (the orchestrator) that is responsible for invoking and combining the single subactivities. The BPEL4WS Spec. (2007) is an example of this approach. BPEL4WS addresses compositions where the control flow of the process and the bindings between services are known in advance.

(2) Web service choreography, instead does not assume a central coordinator but rather defines complex tasks via the definition of the conversation that should be undertaken by each participant; the overall activity is then achieved as the composition of peer-to-peer interactions among the collaborating web services. Moreover, the choreography describes the external visible behavior of the web service. WS-CDL Spec. (2005) is an example of this approach.

5.1.4 Semantic-based approaches. The semantic web (Berners-Lee *et al.*, 2001) allows the representation and exchange of information in a meaningful way, facilitating automated processing of descriptions on the web. Annotations on the semantic web express links between information resources on the web and connect information resources to formal terminologies. These connective structures are called ontologies. Ontologies are used as data models throughout these types of approaches, meaning that all resource descriptions and all data interchanged during service usage are based on ontologies. The extensive usage of ontologies allows semantically enhanced information processing. WSMO Spec. (2005), which is described in Section 2.3 and OWL-S Spec. (2004), which is an OWL service ontology for describing various aspects of web services, are considered as two major approaches based on semantic.

5.2 Comparative evaluation

In this section, we compare the above WSC approaches with respect to the following criteria. It is important to mention that each web service composition approach must fulfill at least a subset of those criteria. In that case, if an approach does address all aspects of a given criterion, the approach quality is marked as "Good". Moreover, an approach is marked as "Average" with respect to a specific criterion, if only a part of what that criterion expects can be fulfilled. Finally, if no aspect of a given criterion fulfill by an approach, the approach is marked as "Low" with respect to that specific criterion. The result can be seen in Table I.

5.2.1 Security. As cited in WS-Sermap (2002), security has been a key factor that was holding companies back from adopting web services. In 2005, the landscape is different especially after the emergence of several specifications for web service security such as WS-Security Spec. (2004), WS-Trust Spec. (2004), and WS-Federation Spec. (2003).

Most of the workflow-based approaches like EFlow neglect consideration of security. Moreover, the major weakness of most of the approaches for service composition based on planning is lack of security mechanism. However, the BPEL4WS specification benefits from WS-Security Spec. (2004) to ensure messages integrity and confidentiality. Nevertheless, these specifications have not found their way to composite web services and among them AI-planning techniques. Owing to the importance of AI-planning techniques for web service composition, their security issues needed to be answered carefully, which is the focus of this paper.

5.2.2 Quality of service. In order to specify non-functional properties of web services, taking QoS into account is one of the hot challenges in the area of web service composition. Majority of web service composition approaches based on workflow do not consider non-functional properties, like EFlow and PPM. Moreover, a planning operator in WSC approaches based on AI-planning, like situation calculus, is not able to address such QoS aspects. Nevertheless, HTN-DL as an AI-planning approach takes

	Criteria						
Approaches	Security	QoS	(Semi) automatic	Scalability	Correctness	Discovery	Overall result
EFlow	N/A	Low	No	Low	Average	No	Low
PPM	N/A	Low	No	Low	Average	No	Low
Situation calculus	N/A	Low	No	Good	Good	No	Average
HTN-DL	N/A	Average	Yes	Good	Good	No	Good
BPEL4WS	Yes	Average	No	Average	Low	No	Average
OWL-S	N/A	Good	Yes	Good	Low	No	Good
WSMO	N/A	Good	Yes	Good	Low	Yes	Good

Table I.
Comparing web service composition approaches

non-functional properties into account by means of ontologies which provide matchmaking in a flexible manner. However, considering non-functional properties cannot be found in all elements of that approach. In addition, BPEL4WS as a web service composition approach based on syntactic cannot support directly QoS aspects. On the other hand, QoS aspects are considered by OWL-S as a semantic-based approach. In that case, these aspects are identified as parameters in definition of web service descriptions. Since functional links between OWL-S metrics cannot be specified, quality conscious service discovery cannot be performed. Eventually, QoS aspects are applied in all definitions of WSMO concepts such as goal, web service, ontology, and mediator. In that case, description of each WSMO concept specifies proper mapping between non-functional properties and that specific element.

5.2.3 Automatic composition. Considering automated composition for web services makes the development of applications faster, makes the user interoperability with complicated set of services easier, and makes the ruse safer. Using automatic composition, the application developer or the end-user can identify the request and then the composition engine is responsible to select appropriate web services and offer the composite service to the requestor. In that case, the way to describe candidate web services, to combine them together, and to specify how much they are matched with the goal is questionable. So far, there is no fully automated approach for composition of web services. However, in this research, majority of the web service composition approaches like, WSMO, OWL-S, and HTN-DL are considered as a semi-automatic approach.

5.2.4 Composition scalability. The capability of a service to deal with several requests in a specific period of time is represented by the composition scalability. It is important to mention that composition of two services are totally different from composition of ten or more services. In a real situation for web service composition, service requestors are willing to interact with a huge number of services. In that case, several hundred web services might be invoked by enterprise applications. Therefore, an important issue which is necessary to be taken into consideration is the scalability of web service composition approaches with the huge number of available services. Normally, the scalability is measured by the number of requests which are addressed by the approach in a given period of time.

Among from workflow-based approaches, the EFlow and PPM which are investigated in this research do not provide tolerable scalability. The situation calculus and HTN-DL, as approaches based on AI-planning, because of applying DLs reasoner, can address a huge number of web services, and thus in this research the HTN-DL and the situation calculus are considered as a web service composition approaches with tolerable scalability. On the other hand, in BPEL4WS as a web service composition approach based on syntactic, due to increasing number of Extensible Markup Language (XML) files, the process of service composition is become tiresome. Since the BPEL4WS is a recursive approach, the composition process is modularized. Finally, the same issues are discussed in WSMO and OWL-S in terms of scalability. In that case, though extra time is needed for synchronization and mapping between XML and class definitions, the approaches have "Good" scalability by definition of classes.

5.2.5 Correctness verifiability. The correctness verifiability can be identified directly based on specifications of web service composition. In that case, the web service composition might be conducted to huge and complicated web service execution systems. The key feature of such systems is their behavior accuracy.

WSC approaches based on AI-planning such as HTN-DL and situation calculus offer powerful verification for determining the correctness. That is, because of the solid mathematical basis of those approaches for evaluating the created plans by the planner. On the other hand, OWL-S as a semantic-based approach provides a rule-based method for composite services by defining control constructs which can increase the verifiability of the generated service. But, no mathematical method is addressed by OWL-S. However, there is no way to evaluate correctness verifiability of the other approaches like EFlow, PPM, BPEL4WS, and WSMO. That is, no direct maintain for verifying web service composition at design time is presented by those approaches. The BPEL4WS, as an instance, concentrates more on implementation instead of specification; therefore, there is no proper solution to offer a formalism for verifying the BPEL4WS flows correctness.

5.3 Discussion

Based on our evaluation, the HTN-DL formalism can be considered as an optimized AI-planning technique. That is because of the following reasons:

- The HTN-DL planning identifies a composite web service to be described by the hierarchical construction of its domains conveniently. Moreover, the key attribute of web services (which is loosely coupled) is adopted with domains of the HTN-DL.
- The HTN-DL provides ontological reasoning for reusing those web services which are identified by different providers with separate contexts in a flexible manner.

Nevertheless, there are still some limitations in HTN-DL that this research tries to address them as follows:

- One of the most prominent problems in HTN-DL which has not been addressed yet is the discovery process of web services. In that case, it is assumed that descriptions of web services are already imported to the system as inputs for the planner. However, that fact is not pragmatic as it is impossible for the planning agent crawling the web and finding all existing services. In this research, that problem is tackled by means of enhanced WSMO discovery process as well as the SCAIMO translator.
- The HTN-DL planning system is based on OWL-S. However, the OWL-S ontology is problematic. For instance, the service profile is the only OWL-S element that can address non-functional properties. However, this research tries to tackle all those restrictions by means of WSMO instead of OWL-S.
- Another problem that has not been considered in that approach is security constraints of web service composition. For instance, a web service provider may not want to proceed with requests which are sent by a specific internet protocol address, or it may want to put some additional security constraints like authentication, authorization, access control, privacy, anonymity, confidentiality, and policy on the composition process. Such constraints must be carefully considered during web service composition. Considering such security issues is also one of the contributions of our work.

To address the following problems, we suggest the SCAIMO framework which is described in the following sections.

6. Security capability and constraint types

This section provides a brief overview on both security capability and constraint concepts. Service capability simply means what the service can do, and they can be used for discovery, advertising, and matchmaking. Security capabilities are defined as a type of service capability to express the security features of a web service, based on the specified security terms. We introduce WSMO security vocabulary for providing those security terms as shown in Figure 3. On the other hand, security constraints are security features which are required. For example, a security constraint can be the X.509 authentication mechanism required by a service requestor.

Security constraints further categorized into three types as follows.

6.1 Security-related goal constraints

Every single requestor requirement must have a corresponding capability on the provider side to satisfy it. These requirements are described as goal constrains and have to be defined in preconditions (which are used to express constrains on inputs,

```
/*****************************
* ONTOLOGY
*****************************/
ontology _"http://www.example.org/ontologies/Security"
    nfp
        dc#title hasValue "WSML example ontology"
        dc#subject hasValue "Security Vocabulary"
        dc#description hasValue "fragments of a security ontology"
        dc#date hasValue _date(2009,7,10)
        dc#format hasValue "text/html"
        dc#language hasValue "en-US"
        wsml#version hasValue "$Revision: 1.1 $"
    endnfp
...
concept privacyAccessControl
hasType impliesType privacyAccessControl

instance P3P memberOf privacyAccessControl
    hasName hasValue "P3P"
instance REI memberOf privacyAccessControl
    hasName hasValue "REI"
instance EPAL memberOf privacyAccessControl
    hasName hasValue "EPAL"
instance XACML memberOf privacyAccessControl
    hasName hasValue "XACML"
...
concept authentication
hasType impliesType authentication

instance WS-Security memberOf authentication
    hasName hasValue "WS-Security"
instance SAML memberOf authentication
    hasName hasValue "SAML"
instance X.509 memberOf authentication
    hasName hasValue "X.509"
```

Figure 3.
An example of security ontology in WSMO

the requestor should be able to provide to the service) of web services description as already shown in Figure 1. For example, a constraint can be a specific encryption algorithm that a user would like to be supported by requested web services.

In the motivating scenario, authentication and encryption are two security-related constraints from the requestor that have to be taken into consideration during discovery and composition processes of web services to fully satisfy the goal.

6.2 Security-related choreography constraints

This category of constraint describes security requirements of web service providers which have to be satisfied by service requestor capabilities. For describing this type of constraint, we use the syntax of WSMO Choreography (Scicluna *et al.*, 2005) with a slight extension, based on control state ASMs. Therefore, only web services will be selected by security matchmaker which can successfully reach the security confirmed state as shown in Figure 4. The figure shows transition rules and states for the orchestration case; for the choreography case exactly the same format and structure is used, therefore it has not been added to the example of Figure 4.

6.3 Security-related orchestration constraint

It can be named as cooperation constraints which refer to conditions that a web service can require from another web service in order to cooperate with. For example, as shown in Figure 4, *BookFligh* web service uses transition rules in WSML notation to describe its security-related orchestration constraints. Then, it only cooperates with *BookHotel* web services which can reach confirmed security state.

7. SCAIMO architecture

In order to realize SCAIMO, 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 of the SCAIMO framework consists of five main components:

(1) *EHTN-DL planner*. This element is considered as the main component of the SCAIMO as it is responsible for composition of semantic web services by means of task decomposition. In that case, there might be composition of two or more methods or operators to fulfill the task request. In order to find those operators or methods which are matched to a specified task, the WSML-DL reasoner is applied by the planner. Moreover, the planner applies the reasoner for determining that applicability by evaluating the preconditions of actions. It is important to mention that in the proposed framework, the planner will not be executed if the user's goal can be satisfied by a single web service. In the next section, more details on the process of planning are presented.

(2) *SCAIMO translator*. This component is used in SCAIMO to translate each WSMO capability of service into an element in the task ontology (T_{ont}) using WSMO ontologies. In addition, the translator provides translating of WSMO service interfaces into a set of methods and operators. This translator is an extension of the AIMO translator (Tabatabaei *et al.*, 2010) which is used in the SCAIMO architecture as well.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { fl   _"http://www.example.org/BookFlight",
            htl  _"http://www.example.org/BookHotel",
            OCRS _"http://ww.example.org/OnlineConferenceRegistrationSystem",
            scr  _"http://www.example.org/Security"}
/******************************
* COMPOSITE WEBSERVICE
*****************************/
webService OCRS_"http://example.org/OnlineConferenceRegistrationSystem"
interface _"http://example.org/OnlineConferenceRegistrationSystem#OCRSInterface"
    choreography
         .....
    orchestration
      stateSignature
         importsOntology { fl#bookFlightOntology, htl#bookHotelOntology,
                           scr#SecurityOntology"}

     state{OCRS#start, OCRS#flightRequested, OCRS#hotelRequested, OCRS#noFlight,
           OCRS#noHotel, OCRS#booked, OCRS#securityCheck, OCRS#securityConfirmed, OCRS#noSecurity}

     transitionRules

      //Request a flight
      if (state = OCRS#start) then
         add (_#fReq[OCRS#date hasValue ?date, OCRS#statrt hasValue ?start,
                     OCRS#destination hasValue ?dest, fl#client hasvalue ?client]
                     memberOf fl#FlightRequest)
         state = OCRS#flightRequested
      endIf

      //No flight available: terminate with failure
      ...
      //Flight offer received: request and Security Check for Book Hotel Service
      if (state = OCRS#flightRequested and
           exists {?fo, fn } (?fo [OCRS#date hasValue ?date, OCRS#statrt hasValue ?start,
                               OCRS#destination hasValue ?dest, fl#flightNumber hasValue ?fn,
                                fl#client  hasValue ?client]
                                memberOf fl#FlightOffer)
      then
          ?flight = ?fo
          add (_HoSecReq[ scr#authentication hasValue ?authentication,
                          scr#encryption     hasValue ?encryption] memberOf scr#SecurityCheck)
          state = OCRS#SecurityCheck
      endIf

      //Security requirement checked: Request Hotel
      ...
      state = OCRS#securityConfirmed
      //No security requirement available: terminate with failure
      if (state = OCRS#SecurityCheck and
          exists {?sna} ?sna [ scr#authentication hasValue ?authentication,
                          scr#encryption     hasValue ?encryption] memberOf scr#SecReqNotAvailable)
      then
        state = OCRS#noSecurity
      endIf

      //Hotel offer received: confirm both flight and hotel and terminate successfully
      ...
      //No hotel available : cancle the flight and terminate with failure
      ...
```

Figure 4.
Security-related choreography and orchestration constrains in SCAIMO

(3) *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. Furthermore, the SCAIMO architecture does take care of non-functional properties during matching the semantic capability descriptions of web services and goals, using discovery component of WSMX.

(4) *Secure task matchmaker*. Secure task matchmaker function is used by WSML-DL reasoner to select for each task a web service satisfying the specified security requirements, which are described in the form of goal choreography and orchestration constraints to achieve security conscious composition. This component is responsible for matching operators and methods based on DLs and WSMO.

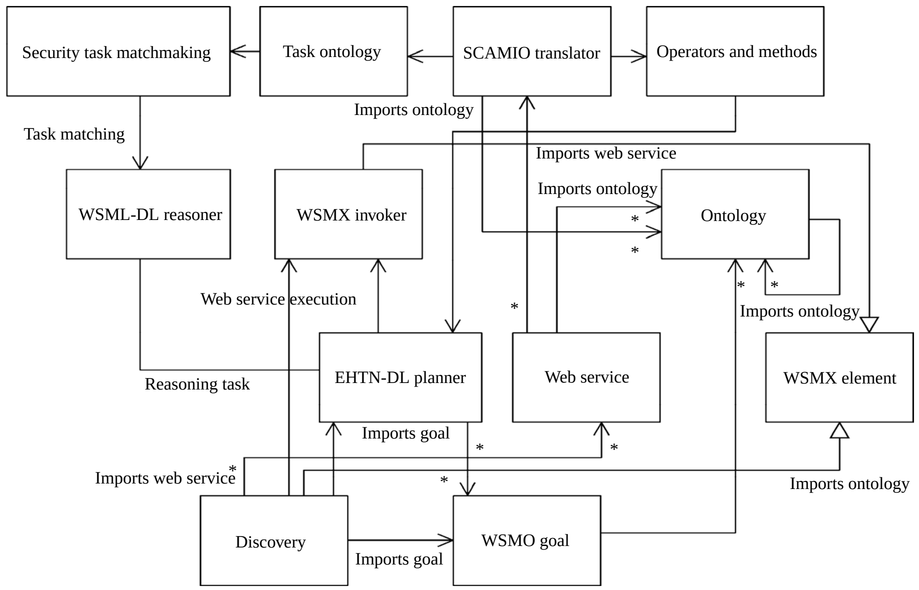


Figure 5.
High-level architecture of the SCAIMO framework

(5) *WSML-DL reasoner*. The expressions of DLs SHIQ (D) are captured by this component which is containing expressions of WSML-DL as a wrapper to the formal syntaxes of DLs (Hustadt *et al.*, 2004). Indeed, this element has ability to provide, among others, reasoning tasks (e.g. instance retrieval, entailment, and ontology consistency checking) for the SCAIMO framework. As mentioned in Fensel *et al.* (2006), though WSML-DL and OWL-DL are semantically similar, yet their ontology models are totally different. In fact, the epistemology which is used by the WSML, conceptualizes from the underlying logical language, but the epistemology of DL is used directly by the OWL. In addition, the logical and the conceptual models for expert and non-expert users have been differentiated by the WSML which cannot be found in OWL. Roughly, the above features of WSML can make it simpler to be used as an ontology language.

The SCAIMO framework mechanism to do both semantic web service discovery and composition by means of WSMO and EHTN-DL planner is described as follows.

In the first step, the WSMO goal is responsible to capture the user's goal as what actually the user would like to obtain. After that, the AIMO tries to find those web services which can satisfy the request by means of enhanced WSMX discovery component. Security constraints of a service requestor can be defined in the form of goal capabilities. They are described in preconditions and explains the requestors constrains which should be satisfied by all candidate services participating in composition. Moreover, non-functional properties of a WSMO goal are defined as a set of attributes which are relating to the request. In addition, the goal mediators provide definitions of goals by reusing previous goals which exist in the WSMO goal repository.

In case of no result after the process of web service discovery to find an atomic service which may fulfill the user's request, the goal is sent to the EHTN-DL planner.

After that, the process of combination of available services is performed by the planner in order to create the process model which can satisfy the goal. Next, WSML-DL reasoner will be queried one by one for the existence of each task. In order to respond to queries, WSML-DL reasoner has to communicate with secure task matchmaker for matching possibilities between the task and operators or methods described by task ontology. In that case, security-related choreography and orchestration constraints of web services will be considered by secure task matchmaker.

With given information from the motivating scenario, we can illustrate the SCAIMO procedure. As described in Section 4, none of the discovered web services can fully satisfy the security-related goal lonely. Therefore, the goal can be achieved by task decomposition using the EHTN-DL planner and the secure task matchmaker. For the sake of simplicity, only composition of two web services namely as *BookFligh* and *BookHotel* is explained. Table II shows candidate services for *BookFligh* and *BookHotel* along with their security capabilities and constraints.

With regards to the constraints explained before, only web services which can satisfy precondition constraints, and security-related goal constraints are selected. In this case, WS1, WS2, WS4, WS5 are capable of satisfying the security goal constraint (supporting of AES encryption). Next, security-related choreography constraints are considered, as the user only supports X.509 authentication, therefore WS2 will be removed from list of candidates. Consequently, the WS1 is the only candidate for *BookFlight* in this stage. In addition, security-related orchestration constraints are considered which refer to conditions that a web service can require from another web service in order to cooperate with. It means that the *BookFlight* web service candidate (WS1) requires to the consequent web services (BookHotel web service candidates) to support X.509 for authentication and DES for encryption. Therefore, in this simple example, WS1 and WS4 are selected for execution path.

8. Implementation of SCAIMO

In order to validate the feasibility of the proposed approach, a prototype called SCAIMO-composer has been developed. The prototype is based on Java, opensource components, and the NetBeans development environment. So far, the SCAIMO-composer provides the following functionalities:

Demanded task	Candidates	Security capabilities	Orchestration constraints	Choreography constraints

BookFlight	WS1	Authentication = X.509
		Encryption = DES, AES
			Authentication = X.509
			Encryption = DES
				Authentication = X.509

	WS2	Authentication = SAML
		Encryption = AES, DES
			Authentication = SAML
			Encryption = DES
				Authentication = SAML

BookHotel	WS3	Authentication = WS-security
		Encryption = DES		

	WS4	Authentication = X.509
		Encryption = AES, DES
				Authentication = X.509

	WS5	Authentication = SAML
		Encryption = AES, DES		

Table II. Security capabilities and constraints of web services

- Translating the web service descriptions from WSDL into WSMO in order to be adapted with the discovery component of WSMX which is described in Section 7.
- Discovering the web services through the way of manipulating the relations between ontologies and based on the functionalities, non-functional properties, and security constraints which are specified by the user as a goal at process run-time. An example of this process can be shown in Figure 6.
- Transformation of the discovered web services to EHTN-DL using the proposed AIMO translator which is shown in Figure 7. More details on AIMO translator can be found in Tabatabaei *et al.* (2010).
- Composing the discovered web services based on the proposed algorithm using EHTN-DL formalism.

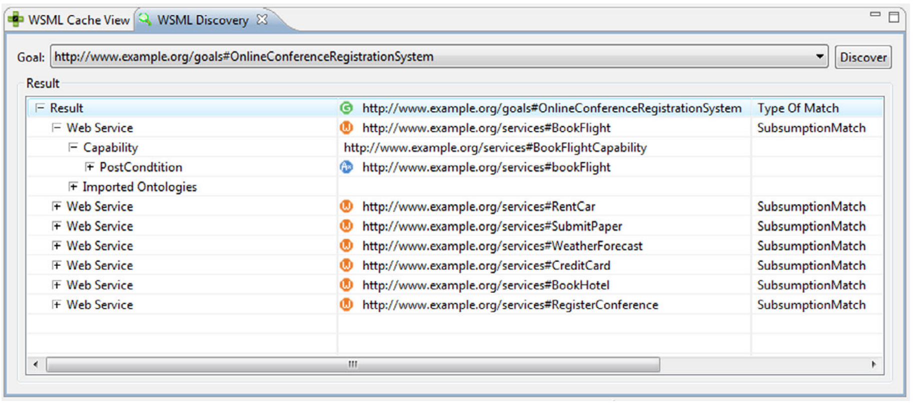
The remainder of this section describes that how the motivating scenario can be realized using the SCAIMO-composer.

In the motivating scenario, the security-related goal is OCRS. As described in Section 7, the first step is discovering proper web services which are matched to the goal. We do address that process using web service modeling toolkit (WSMT) (WSMO Spec., 2005). Moreover, security-related choreography constraints (which are described in Section 6) will be taken into consideration in this step. The result of this stage can be shown in Figure 6. As this figure shows, seven web services (like *BookFlight*, *BookHotel*, *RentCar*, and so on) can fulfill the request. However, none of discovered web services can fully satisfy the user's goal as the matching type is subsumption for all of them.

Therefore, the EHTN-DL formalism which is described in Section 7 has to be used to generate a proper plan for combining the discovered web services. In that case, the AIMO-translator is responsible to provide acceptable web services formats for the composer as shown in Figure 7. Note that the planner does take care of security-related orchestration constraints during the planning process as described in Section 6.

9. Experimental evaluation

The experiment has been designed in a way to show the performance and the scalability of our approach for web service composition in the context of the motivating scenario.

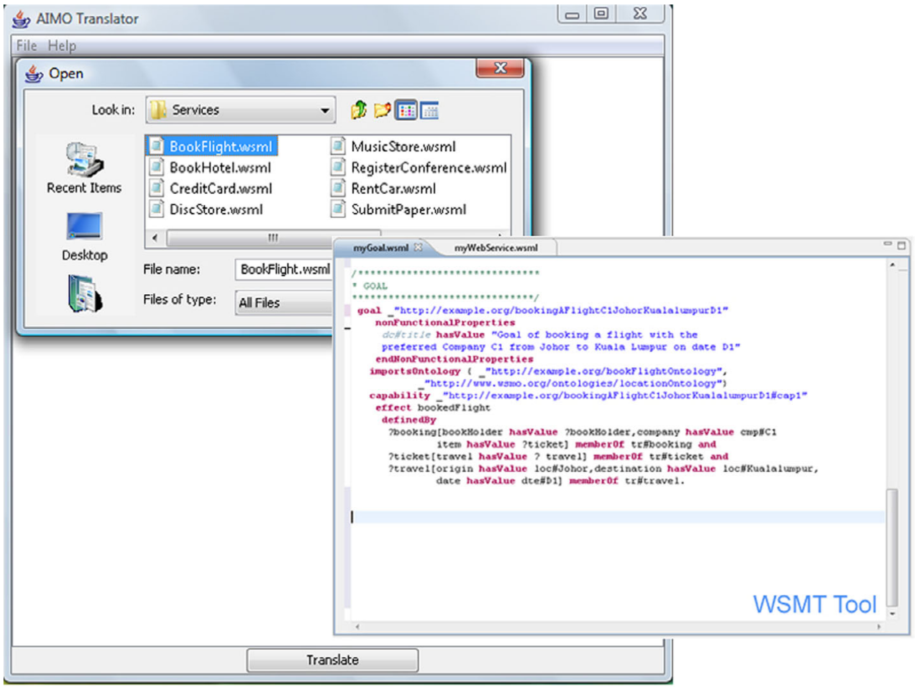


Goal: http://www.example.org/goals#OnlineConferenceRegistrationSystem

Result

Result	http://www.example.org/goals#OnlineConferenceRegistrationSystem	Type Of Match
Web Service	http://www.example.org/services#BookFlight	SubsumptionMatch
Capability	http://www.example.org/services#BookFlightCapability	
PostCondtition	http://www.example.org/services#bookFlight	
Imported Ontologies		
Web Service	http://www.example.org/services#RentCar	SubsumptionMatch
Web Service	http://www.example.org/services#SubmitPaper	SubsumptionMatch
Web Service	http://www.example.org/services#WeatherForecast	SubsumptionMatch
Web Service	http://www.example.org/services#CreditCard	SubsumptionMatch
Web Service	http://www.example.org/services#BookHotel	SubsumptionMatch
Web Service	http://www.example.org/services#RegisterConference	SubsumptionMatch

Figure 6.
Discovery process for the motivating scenario using WSMT tool



```
/*****************************
* GOAL
*****************************/
 goal _"http://example.org/bookingAFlightC1JohorKualalumpurD1"
    nonFunctionalProperties
     dc#title hasValue "Goal of booking a flight with the
     preferred Company C1 from Johor to Kuala Lumpur on date D1"
    endNonFunctionalProperties
   importsOntology ( _"http://example.org/bookFlightOntology",
          _"http://www.wsmo.org/ontologies/locationOntology")
   capability _"http://example.org/bookingAFlightC1JohorKualalumpurD1#cap1"
    effect bookedFlight
     definedBy
      ?booking[bookHolder hasValue ?bookHolder,company hasValue cmp#C1
              item hasValue ?ticket] memberOf tr#booking and
      ?ticket[travel hasValue ? travel] memberOf tr#ticket and
      ?travel[origin hasValue loc#Johor,destination hasValue loc#Kualalumpur,
              date hasValue dte#D1] memberOf tr#travel.
```

Figure 7.
AIMO translator prototype to support SCAIMO

In order to evaluate the proposed approach, 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 part of our web services repository. Totally, our repository includes 250 semantically annotated web services like *BookHotel* and *CreditCardCheck*. All the experiments presented in this section have been performed on a Pentium 4 CPU 2.66 GHz computer with 1.0 GB memory using JDK 1.6.0. For evaluation of the proposed approach using the repository above and some other services that have been added to the repository, we have used the motivating scenario described in Section 3. Therefore, we have measured the execution time for discovery and composing of seven services to show the performance and the scalability of the approach. The goal is to show to which extent the proposed approach can find all plans for large composition problems in a timely manner. Therefore, we have measured the time for different repository size and number of users. The unit of measurement of composition time is millisecond.

As a result, the experiments can be affected by the order of the web services in the repository, we have randomly replaced the web services in the repository and repeated the experiment for each repository size and number of users for specific number of times. In the following section, we do calculate the replication number for our experiment.

9.1 Determining replication number or sample size

As it is very complex to consider the order of web services in the repository as a factor, therefore we have to replicate the test to increase the precision. Equation (1) has been used for determining the best number of times to repeat the experiment for the same combination factors:

Determining replication number:

$$n \geq \left(1.96 \times \left(\frac{\alpha}{\omega}\right)^2\right) \quad (1)$$

If repository with different orders of services can be considered as new sample, then replication number can be also equal to the sample size. As composition time for a single user and registry of 50 services is approximately 6,100 milliseconds. Therefore, if the estimation of 0.1 percent is desirable for us, then ω is 6.1. On the other hand, as we know the ordering of the services in registry has randomized, therefore even the small number of replications can give us a good estimation. Based on that knowledge, we checked the standard deviation from $n = 2$ to 10 times replication and from $n \geq 6$ the standard deviation of the sample was around 7.1 milliseconds with the error estimation of 0.001. Finally, based on equation (1), n can be 5 or greater. Consequently, we adopted the $n = 5$ for our experiments.

As mentioned, the experiments conducted for five populations of web service registry size included 50, 100, 150, 200 and 250 and for five populations of users included 1, 2, 3, 4, and 10 users. In this case, comparison of composition time of more than two independent samples (as knowing of any of them occurs does not affect in our probability calculation of the other one) is our interest. First, we show the collected results in a form of diagram to have general and basic ideas of trends. Since we have many experiments, the most proper and understandable diagram is the box plot. The diagram is shown in Figure 8.

Furthermore, the test for scalability can be broken into two tests as follows (these tests can be interpreted as a basis for our hypothesis):

- Are there any differences between composition times when number of web services vary from 50 to 250 for different number of concurrent users in the system?
- Are there any differences between composition times when number of concurrent users vary from one to ten for different repository sizes?

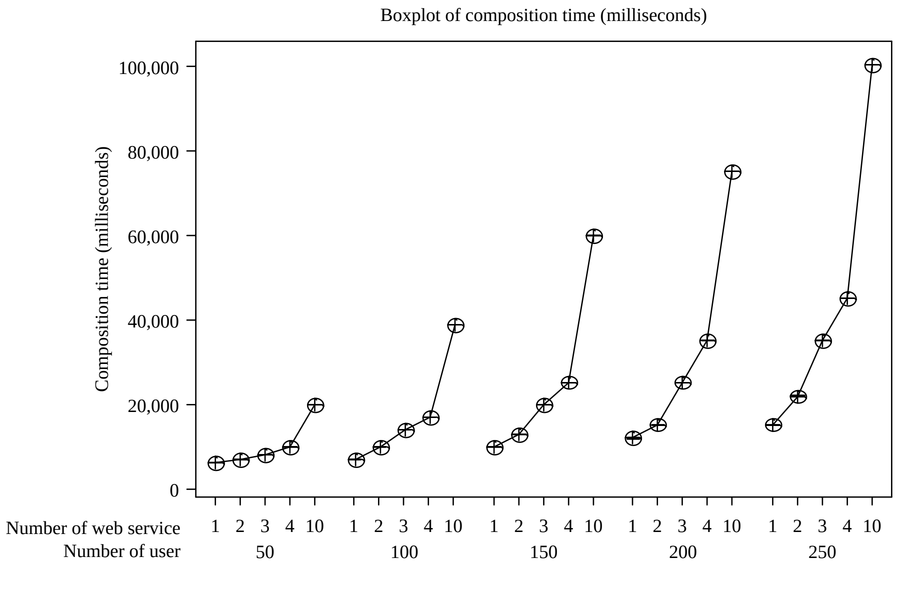


Figure 8.
Boxplot of web service composition time

9.2 Analysis of variance test

If one single test has to be considered, analysis of variance (ANOVA) can be considered as a good option. However, two assumption of normality for residuals and constant variance (by plotting the standardized residuals against the fitted values) have to be checked. For ANOVA, the hypothesis can be written as follows. The null hypothesis suggests no differences in means of composition times for different repository size. Moreover:

H0. $\mu_1 = \mu_2 = \cdots = \mu_5$.

H1. *H0* is not true.

9.2.1 ANOVA result analysis. For each experiment, we assign a symbol of (x, y). x determines the number of users and y determines the number of web services. Therefore, $x \in X = \{1, 2, 3, 4, 10\}$ and $y \in Y = \{50, 100, 150, 200, 250\}$.

We start with analyzing of changes in composition time when the number of web services changes in the repository for various numbers of users. After that, we shift into analysis of composition time versus the number of users for different number of web services. In addition, for all ANOVA test two assumption of normality for residuals and constant variance (by plotting the standardized residuals against the fitted values) were checked and result shows that assumptions were correct for all ANOVA tests.

9.2.2 Composition time versus number of web services analysis. First, all groups have *p*-value of <0.05 which shows strong evidence that there are differences in mean composition time between five repository sizes. First test was done on (1, Y) which shows the mean of composition time has been increased from 6,093 to 15,121. And the highest jump is reported for scaling up to 250 web services where a 95 percent confidence interval (CI) for differences in means of composition time is (9,016, 9,039). However, for a single user scaling up to 100 and 150 still reasonable CIs have been reported.

It is also worthy to know that the highest rise in composition times for two consecutive registry size on the graph has been reported for 200 and 250 which has 95 percent CI of (3,094.15, 3,115.85). Therefore, it can be recommended for a single user, it is best not to scale up to >200 size registry.

Second ANOVA table is for (2, Y) case when there are two users in systems and we analyze effects of changes in registry size on the composition time. It shows the mean of composition time has been increased from 7,013 (for registry size of 50) to 35,051.2 (for registry size of 250) which compare to the (1, Y) case shows a considerable increase in composition time for 250 web services.

That shows entering another user to the single user system notably intensifies rise in composition time for scaling up to 250 size. Furthermore, the highest rise in composition times for two consecutive registry sizes on the graph has been reported for 200 and 250. In general, in all mean difference, there is a considerable increase compared to (1, Y).

Third ANOVA table of (3, Y) shows 95 percent CI for differences in means of composition time is (11,970.4, 11,977.3) for scaling up from 50 to 150. That shows very interesting trend. If number of users in the system increases, then scaling up becomes more costly in terms of time. Consequently, even that scale up is more costly compare to scaling up from 50 to 250 sizes in single user case. Therefore, for scaling up two factors of number of users and repository size have to be considered simultaneously. ANOVA for (4, Y) also shows the same trend and suggest the both factor consideration for scaling up. In addition, this time even a scaling up to 100-95 percent CI for mean differences of (7,068.9, 7,086.3) – from 50 has the same effect on the composition compare to the scale from 50 to 250 in (1, Y). This can clearly shows number of user in the system has higher effect on determining the composition time. The previous claim can be verified by considering (10, Y) as even a scale up of 50-100 is very costly for the system.

9.2.3 Composition time versus number of users analysis. First test was done on (X, 50) which shows the mean of composition time has been increased from 6,093.4 to 20,135.2. And the highest difference is reported for scaling up to ten users where a 95 percent CI for differences in means of composition time reported 14,033 and 14,050.7. Therefore, analysis clearly shows by increasing users from one to ten the composition time increased more compare to the time the repository size changed from 50 to 250 in single user case, which again suggest that number of user in the system has higher effect on composition time.

ANOVA test result can be very helpful for the system administrator. For example, consider the second case of (X, 100). Assume that administrator is dealing with two users in a system and he has adhere to a service level agreement (SLA) of not having composition time of more than 13,000 ms or pay compensation of violating SLA to the users. In this case considering the ANVOA result, if third user comes to the system then a 95 percent CI for differences in means of composition time becomes (6,944.7, 6,963.3). Therefore, he can calculate the compensation amount and compare it with benefit coming out of serving the third user and finally decide whether to accept the new user or not.

The test on (X, 250) reveals that this case has the greatest differences between means of composition times among various number of concurrent users. For example, a scale up of single user system to system with three user lead to raise differences in means of composition time form approximately (with 95 percent of CI) 19,921 to 19,939.

9.3 Testing correlation

Pearson correlation test has been done for all groups and p was under 0.05 which means the "H_0 = there is no correlation" is not true. The value found for p for each group has been reported in Table III.

As it can be deducted by analyzing the data in the table, the strength of existing linear association between user number and composition time and also number of web services and composition time is high.

9.4 Multiple regression and single variable regression

Here, the multi-predicator regression (for number of web services and number of users) has been used and the normality for residuals and constant variance assumption has been tested. As it can be evidenced by Figure 9, the assumption checks are not passed and, therefore, we cannot rely on the result. Transformations of many types (multiplication of factors and response, square root, addition or subtraction by constants, log ten, log of log, etc.) have been tested and none of them could help in satisfaction of assumptions.

However, as we tried groups like (1, Y) for one factor regression (Figure 10) by a transformation of log(log(log(composition time))), therefore distribution of residuals

Table III.
Correlation

Group	(1, Y)	(2, Y)	(3, Y)	(4, Y)	(10, Y)	(X, 50)	(X, 100)	(X, 150)	(X, 200)	(X, 250)
Correlation	0.989	0.974	0.992	0.997	0.998	0.996	1.000	0.999	0.997	0.997

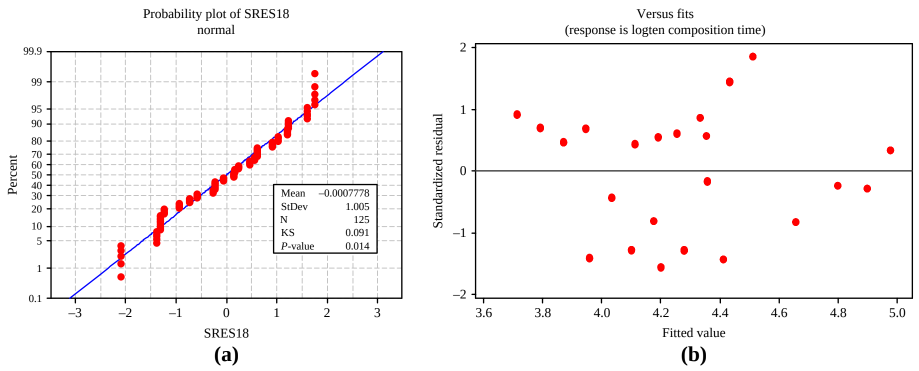


Figure 9. Two predictor assumption checking

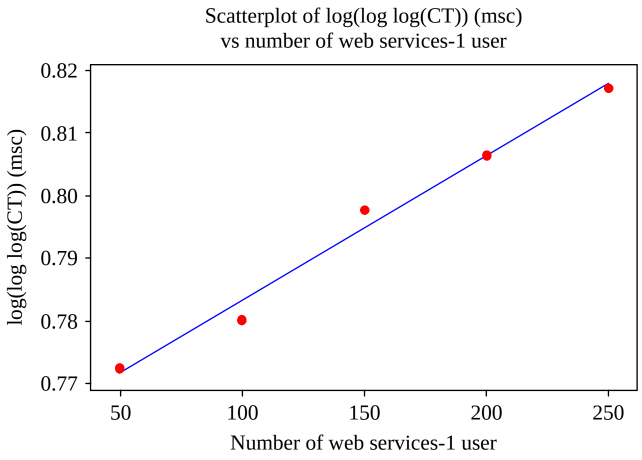


Figure 10. Linear model (1, Y)

change to normal ($p = 0.15$) and the variance was constant. Figure 11 shows the regressing models for (1, Y). As it can be seen, in quadratic model 4 out of 5 dots located inside 95 percent CI.

In addition, for (X, 50) without transformation we can get the regression equation as can be seen in equation (2):

Regression:

$$\text{Composition time (millisecond)} = 3{,}823 + 1{,}609 \text{ number of users (for 50 web services)} \quad (2)$$

The regression concludes that when we have 50 web services in repository there is a statically significant association between composition time and number of users in system. The regression model is shown in Figure 12. Consequently, for a system with one user and 50 web services, we can conclude number of users is statically more important predicators.

10. Conclusion and future work

In this paper, we have summarized ongoing work on the development of SCAIMO. In contrast to most works in AI-planning web service composition, our approach

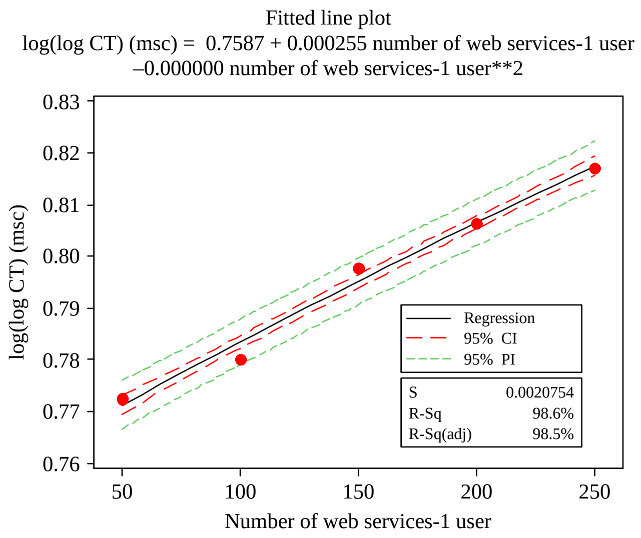


Figure 11.
Quadratic model for (1, Y)

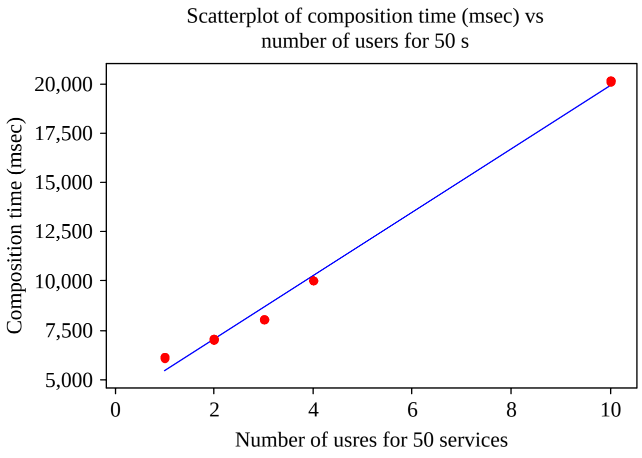


Figure 12.
Linear model for (X, 50)

concerns about security issues. The SCAIMO framework takes care of security requirements of both web service requestors and providers using secure task matchmaker which is responsible of matching tasks with operators and methods. In order to validate the feasibility of the proposed approach, a prototype called SCAIMO-composer has been developed. Moreover, the SCAIMO is evaluated with respect to performance and scalability, and results show that the proposed approach is effective and efficient.

There are several directions for future work to further enhance SCAIMO features. Future research can look into other non-functional constraints and capabilities in WSC such as reliability, persistence and trust for the proposed architecture.

References

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P. (2003), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, San Francisco, CA.

Berners-Lee, T., Fielding, R. and Masinter, L. (1998), "Uniform resource identifiers (URI): generic syntax", IETF RFC 2396, available at: www.ietf.org/rfc/rfc2396.txt

Berners-Lee, T., Hendler, J. and Lassila, O. (2001), "The semantic web", *Scientific American*, Vol. 284 No. 5, pp. 34-43.

BPEL4WS Spec. (2007), "Business process execution language for web services version 1.1", available at: www.ibm.com/developerworks/library/specification/ws-bpel/

Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V. and Shan, M. (2000), "Adaptive and dynamic service composition in EFlow", *Proceedings of the International Conference on Advanced Information Systems Engineering* (*CAiSE*), *Stockholm*, LNCS 1789, Springer, Berlin, pp. 13-31.

Fensel, D. and Bussler, C. (2002), "The web service modeling framework WSMF", *Electronic Commerce Research and Applications*, Vol. 1 No. 2, pp. 113-37.

Fensel, D., Lausen, H., Polleres, A., De-Bruijn, J., Stollberg, M., Roman, D. and Domingue, J. (2006), *Enabling Semantic Web Services: Web Service Modeling Ontology*, Springer, Heidelberg.

Fikes, R.E. and Nilsson, N.J. (1990), *Strips: A New Approach to the Application of Theorem Proving to Problem Solving*, Kaufmann, San Mateo, CA.

Ghallab, M., Nau, D. and Traverso, P. (2004), *Automated Planning: Theory and Practice*, Morgan Kaufmann, Amsterdam.

Hakimpour, F., Sell, D., Cabral, L., Domingue, J. and Motta, E. (2005), "Semantic web service composition in IRS-III: the structured approach", *Proceedings of the IEEE International Conference on E-Commerce Technology* (*CEC'05*), *Manchester, Germany*, pp. 484-7.

Horrocks, I., Sattler, U. and Tobies, S. (2000), "Practical reasoning for very expressive description logics", *Logic Journal of the IGPL*, Vol. 8 No. 3, pp. 239-63.

Hustadt, U., Motik, B. and Sattler, U. (2004), "Reducing SHIQ-description logic to disjunctive Datalog programs", *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning* (*KR 2004*), AAAI Press, Menlo Park, CA, pp. 152-62.

Kuter, U., Nau, D., Pistore, M. and Traverso, P. (2005), "A hierarchical task-network planner based on symbolic model checking", *Proceedings of the International Conference on Automated Planning and Scheduling* (*ICAPS'05*), AAAI Press, Menlo Park, CA, pp. 300-9.

McIlraith, S. and Son, T.C. (2002), "Adapting Golog for composition of semantic web services", *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning* (*KR'02*), *Toulouse*, pp. 482-93.

Medjahed, B., Bouguettaya, A. and Elmagarmid, A.K. (2003), "Composing web services on the semantic web", *The International Journal on Very Large Data Bases* (*VLDB*), Vol. 12 No. 4, pp. 333-51.

OWL-S Spec. (2004), *OWL-S: Semantic Markup for Web Services*, W3C Member Submission, available at: www.w3.org/Submission/OWL-S/

Rao, J., Kungas, P. and Matskin, M. (2004), "Logic-based web services composition: from service description to process model", *Proceedings of the IEEE International Conference on Web Services* (*ICWS 2004*), *San Diego, CA*, pp. 446-53.

Rao, J., Dimitrov, D., Hofmann, P. and Sadeh, N. (2006), "A mixed initiative approach to semantic web service discovery and composition: SAP's guided procedures framework", *Proceedings of the IEEE International Conference on Web Services* (*ICWS 2006*)*, Chicago, IL*, pp. 18-22.

Scicluna, J., Polleres, A., Roman, D., Feier, C. and Fensel, D. (2005), "Ontology-based choreography and orchestration of WSMO services", Working Paper d14v0.2, WSMO, available at: www.wsmo.org/TR/d14/v0.2/20050702/

Shuster, H., Georgakopoulos, D., Cichocki, A. and Baker, D. (2000), "Modeling and composing service-based and reference process-based multi-enterprise processes", *Proceedings of the International Conference on Advanced Information Systems Engineering* (*CAiSE*), LNCS 1789, pp. 247-63.

Sirin, E. (2006), "Combining description logic reasoning with AI planning for composition of web services", PhD thesis Department of Computer Science, University of Maryland, College Park, MD.

Sirin, E., Parsia, B. and Hendler, J. (2005), "Template-based composition of semantic web services", *Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web, Arlington, VA*, pp. 85-92.

Sirin, E., Parsia, B., Wu, D., Hendler, J. and Nau, D. (2004), "HTN planning for web service composition using SHOP2", *Journal of Web Semantics*, Vol. 1 No. 4, pp. 377-96.

Steinmetz, N. (2007), "Monitor of the implementation process of the reasoner modules", available at: http://tools.deri.org/wsml/OverviewReasoner.html

Tabatabaei, S., Wan-Kadir, W. and Ibrahim, S. (2008a), "A comparative evaluation of state-of-the-art approaches for web service composition", *Proceedings of the International Conference on Software Engineering Advances* (*ICSEA'08*)*, Malta*, IEEE CS Press, Sliema, pp. 488-93.

Tabatabaei, S., Wan-Kadir, W. and Ibrahim, S. (2008b), "Semantic web service discovery and composition based on AI-planning and web service modeling ontology", *Proceedings of the IEEE Asia-Pacific Services Computing Conference* (*IEEE APSCC'08*)*, Taiwan*, IEEE CS Press, Taipei, pp. 397-403.

Tabatabaei, S., Wan-Kadir, W., Ibrahim, S. and Vahid-Dastjerdi, A. (2010), "AIMO translator: bridging the gap between semantic web service discovery and composition", *Proceedings of the International Conference on Internet and Web Applications and Services* (*ICIW 2010*)*, Spain*, IEEE CS Press, Taipei, pp. 268-73.

Web Services Architecture Requirements (2004), W3C Working Group Note, available at: www.w3.org/TR/wsa-reqs/

WS-CDL Spec. (2005), "Web services choreography description language version 1.0", W3C Candidate Recommendation, available at: www.w3.org/TR/ws-cdl-10/

WS-Federation Spec. (2003), *Web Services Federation Language Version 1.1*, available at: www.ibm.com/developerworks/library/specification/ws-fed/

WSML Spec. (2005), *Web Service Modeling Language* (*WSML*), W3C Member Submission, available at: www.w3.org/Submission/WSML/

WSMO Spec. (2005), *Web Service Modeling Ontology* (*WSMO*), W3C Member Submission, available at: www.w3.org/Submission/WSMO/

WSMX Spec. (2005), *Web Service Execution Environment* (*WSMX*), W3C Member Submission, available at: www.w3.org/Submission/WSMX/

WS-Security Spec. (2004), *Web Services Security: SOAP Massage Security Version 1.0*, OASIS Standard, available at: www.oasis-open.org/specs/#wssv1.0

WS-Sermap. (2002), *Security in a Web Services World: A Proposed Architecture and Roadmap*, IBM and Microsoft white paper, available at: www.ibm.com/developerworks/library/specification/ws-secmap/

WS-Trust Spec. (2004), *Web Services Trust Language*, available at: www.ibm.com/developerworks/library/specification/ws-trust/

About the authors

Sayed Gholam Hassan Tabatabaei is a PhD candidate at the Universiti Teknologi Malaysia (UTM) where he is studying Software Engineering. He obtained his Bachelor's and Master's degrees in Software Engineering from the IAU University of Iran in 2003 and 2006, respectively. He was the Head of the Department of Computer Engineering at the IAU University-Behbahan branch. Software engineering, web services, semantic web and web intelligence are among his research interests. He intends to broaden his perspectives in interdisciplinary fields towards a career in software engineering and web intelligence. Sayed Gholam Hassan Tabatabaei is the corresponding author and can be contacted at: sayed@ic.utm.my

Amir Vahid Dastjerdi is currently a PhD student in Clouds Lab at the University of Melbourne. He is pursuing his research in service deployment in Cloud under supervision of Professor Rajkumar Buyya.

Wan M.N. Wan Kadir is an Associate Professor in the Software Engineering Department, Faculty of Computer Science and Information Systems, UTM. He received his BSc from Universiti Teknologi Malaysia, MSc from UMIST and PhD in the field of Software Engineering from The University of Manchester. He has been an academic staff at Software Engineering Department for more than ten years, and he was the Head of the Department from 2005 to 2009. He is the Chairman of the 2nd Malaysian Software Engineering Conference (MySEC'06), and a member of pro-tem committee of Malaysian Software Engineering Interest Group (MySEIG). He serves as a Program Committee member of the 5th, 4th, and 3rd International Conference on Software Engineering Advances (ICSEA 2010, ICSEA 2009, ICSEA'08), the 15th and 16th Asia-Pacific Software Engineering Conference (APSEC 2009, APSEC 2008), the 5th and 4th International Conference on Software and Data Technologies (ICSOFT 2010, ICSOFT 2009), the 5th and 4th International Conference on Novel Approaches in Software Engineering (ENASE 2010, ENASE 2009), the 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2008), and the 3rd and 4th Malaysian Software Engineering Conference (MySEC'07 and MySEC'08). Most of the proceedings are published by IEEE. His research interest covers various SE knowledge areas based on the motivation to reduce the cost of development and maintenance as well as to improve the quality of large and complex software systems.

Suhaimi Ibrahim is an Associate Professor at the Centre for Advanced Software Engineering (CASE), Faculty of Computer Science and Information Systems, UTM. He is currently appointed as the Deputy Director of CASE and is involved in several short terms and National research schemes of software development, software testing and maintenance projects. He is an ISTQB certified tester of foundation level and currently being appointed as a board member of the Malaysian Software Testing Board. He is actively involved in syllabus and curriculum review of software engineering at the bachelor and post-graduate levels. His research interests include requirements engineering, web services, software process improvement and software quality.

Elahe Sarafian has been Manager of the Planning and Project Control Department at a heavy machineries company in Iran. Her research interest mainly concerns SOA application development for resource management in industry.