A Refined Goal Model for Semantic Web Services

Michael Stollberg
Digital Enterprise Research Institute
University of Innsbruck
6020 Innsbruck, Austria
Email: michael.stollberg@deri.org

Barry Norton
Knowledge Media Institute
Open University
Milton Keynes, MK7 6AA, UK
Email: b.j.norton@open.ac.uk

***Abstract*— The idea of service orientation envisions dynamic detection and execution of suitable Web services for solving a particular request. Most realization approaches pay only little attention to the client side of such architectures. We therefore promote a goal-driven approach: a client merely specifies the objective to be achieved in terms of a goal, and the system resolves this by automated detection, composition, and execution of Web services. Extending the WSMO framework, we present a model for describing goals as formalized client objectives that carry all information relevant for automated detection and execution of Web services. This paper explains the design of the goal model, specifies the formal descriptions of goals, and demonstrates the model within an illustrative example.**

I. Introduction

To overcome the deficiencies of monolithic architectures with regard to reusability and interoperability, the emerging concept of service-oriented architectures (SOA) envisions dynamic detection and execution of Web services for solving a particular request [8]. Most realization approaches focus on the description and handling of Web services but pay little attention to the client side, i.e. request formulation by clients and their interaction with the system.

Following prior conceptions for automated problem solving, we therefore proclaim a goal-driven approach. A client shall formulate the objective to be achieved in terms of a goal, and the system solves this by automatically discovering, composing, and executing appropriate Web services on basis of formal, declarative descriptions. The aim is to enable problem-oriented usage of Web services: the client can concentrate on the problems to be solved while all details on the automated usage of Web services are handled by the system.

This paper presents a description model for goals as the central notion of this approach. As a revision of the goal model developed in the DIP project (`dip.semanticweb.org`, [21]), we define the following extensions for the initial goal model of the Web Service Modeling Ontology WSMO (`www.wsmo.org`): (1) the differentiation of *goal templates* as generic objective descriptions and *goal instances* as concrete requests that instantiate a goal template, (2) automated Web service usage by *client interfaces* which are embedded in mediators that connect Web services and goals, and (3) *composite goals* as an orchestrated workflow of goals that the client desires. We shall explain and demonstrate that this model provides a proper basis for realizing goal-driven Web service usage in service-oriented architectures.

The paper is organized as follows. Section II introduces our goal model with respect to arising requirements and the initial WSMO approach. Section III specifies the semantic descriptions and correlations of the goal model elements, and Section IV demonstrates this in an example from the travelling domain. Finally, Section V concludes the paper and discusses related work.

II. Requirements and Approach

The following first discusses the requirements arising on goal descriptions for realizing goal-driven Web service usage. After recalling the relevant aspects of WSMO wherein our approach is embedded, we then introduce the goal model that is subsequently specified in the remainder to the paper.

A. Requirements Analysis

The central notion in our context is a *goal* as the formal specification of a client's objective that shall be achieved by the usage of Web services. As [21] discusses in detail, three requirements arise on goal descriptions in order to properly support goal-driven Web service usage as outlined above.

1) Expressiveness for Objective Specification: the goal description model must be sufficiently rich for specifying all kinds of objectives that can be achieved with Web services. Goals should be (1) formulated on the knowledge level, i.e. on the level of problems to be solved without any technical details that are irrelevant for the objective to be achieved, and (2) allow to specify restrictions on the workflow of how a goal shall be resolved.

2) Automated Web Service Usage: a goal must carry all information required for automated discovery, composition, and execution of Web services. In particular, these are:

- all inputs required for invoking and consuming the Web services that are to be used
- a compatible counterpart for the communication behavior of the Web service for consuming its functionality
- support for the technical grounding of the Web services for information exchange.

3) Ease of Goal Formulation: practical experiences reveal that most end-users are not capable of specifying complex formal descriptions. Hence, goal specification should be left to knowledge engineers with support for goal formulation by end-users via graphical interfaces.

B. Web Service Modeling Ontology WSMO

The Web Service Modeling Ontology WSMO is a comprehensive framework that envisions semantically enabled service-oriented computing (see [9] for an extensive overview). It defines description models and languages four top-level notions: *ontologies* that specify formalized domain knowledge used in all other elements, *Web services* that carry a semantic description, *goals* as the client objectives to be solved, and *mediators* that resolve potentially occurring mismatches. Relevant in our context, we briefly recall the descriptions of Web services and goals.

A WSMO Web service description consists of two central parts. At first, the *capability* describes the overall functionality provided by a Web service in terms of preconditions, assumptions, postconditions, and effects; these are logical expressions, specified e.g. in WSML [3]. Secondly, two types of interfaces describe the interaction behavior support by a Web service: the *choreography* is the interface for consumption of the Web service by a client, and the *orchestration* is the interface for interacting with other Web services that are aggregated to achieve the overall functionality. Both can be described by ontologized Abstract State Machines [18], [20].

A WSMO goal description consist of a *requested capability* and *requested interfaces*. The former shall specify the objective to be achieved in terms of a capability from the client perspective. The latter is intended to specify the communication behavior for automated Web service usage supported and required by the client. However, the distinction of choreography and orchestration descriptions in the goal model remains ambiguous, and it does not encompass a formal notion for goal instantiations by clients. Thus, the initial WSMO goal model appears to be not sophisticated for realizing goal-driven Web service usage as outlined above.

C. Goal Model Overview

On basis of the experiences and insights gathered in the DIP project, we revise and extend the initial WSMO goal model to consists for four notions. Figure 1 gives an overview as a UML class diagram. To meet the determined requirements, the central changes are the following.

1) Goal Templates and Goal Instances: we distinguish two types of goals. A *goal template* is a generic objective description that is defined at design time and is kept in the system. A *goal instance* denotes a concrete request of a client that is defined at runtime by instantiating a goal template with concrete inputs. Successfully applied in WSMO-enabled systems such as IRS [4] and SWF [23], this distinction allows (1) to support goal instance formulation by end-users via graphical user interfaces (*cf* third requirement), and (2) to allocate expensive operations for Web service discovery and composition on the level of goal templates, therewith improving the runtime efficiency of the SOA system.

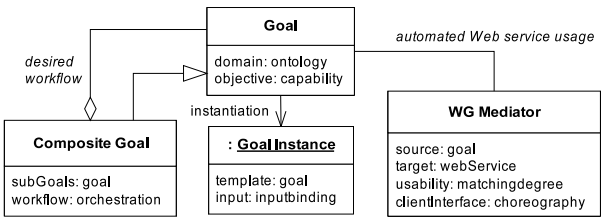

Fig. 1. Goal Model Overview

2) Two Goal Template Types: we further distinguish two types of goal templates with respect to the expressiveness of goals for objective specification (*cf* first requirement). As the basic type, a *goal* is the generic description of an objective that defines conditions on the start state of the world and conditions on the desired state wherein the objective is considered to be satisfied. We specify this in terms of a WSMO capability. A *composite goal* extends this with a desired workflow that shall be sustained during the resolution of the goal. This is a collection of subgoals with control- and data flow among them, which we describe as a WSMO orchestration. The goal template type is independent of the need for Web service discovery or composition for goal solving.

3) Automated Web Service Usage via WG Mediators: in general, there can be several Web services that are usable for solving a goal. Each Web service may provide a different interface for consuming its functionality. We do not consider this to be relevant for goal formulation on the knowledge level. Hence, we allocate the relevant information for automated Web service usage (*cf* second requirement) in a *client interface* [7]. Essentially, this is the compatible counterpart of the Web service choreography interface that allows to interact with the Web service for consuming its functionality. This is allocated in a WSMO *WG mediator* that connects a goal template with a usable Web service. A WG mediator is defined for each Web service that is usable for resolving the goal template.

III. Goal Model Specification

We now turn towards the definition and formal description of the four elements of the goal model. Following the conception in related AI research (e.g. [2], [17]), we understand a goal as the formal description of a client's desire to get from the current state of the world to a state wherein the objective is satisfied. In accordance to the conception of WSMO, the primary focus of our model are the functionalities provided by Web services and requested by goals.

We specify the meaning of goal descriptions within *Abstract State Spaces* (short: ASS) [13]. This defines a state based model of the world that Web services act in with precisely defined formal semantics. Therein, a particular execution of a Web service or a composition of Web services denotes a sequence of state transitions $\tau = (s_0, \ldots, s_m)$, i.e. a change of the world from a start state s_0 to a termination state s_m that is triggered by the invocation with concrete inputs. For solving a goal, we need to find a Web service or a composition that can be invoked such that the client objective is solved in the termination state of the triggered execution.

A. Goals

As the basic goal template type, a goal formally describes an objective with respect to the start- and the desired final state of the world. In service-oriented architectures, the current state of the world may not be known because the constituting facts are distributed and possibly not accessible to all parties involved in the interaction. Hence, we describe a goal $\mathcal{G}$ by conditions on the possible start states and conditions on the desired state of the world wherein the objective is considered to be solved. Formally, this is described by WSMO capabilities that are extended with notions from the ASS-model.

A WSMO capability is defined by *preconditions* ϕ^{pre} as constraints on the information space before a Web service can be executed, *assumptions* ϕ^{ass} as conditions that need to hold in every state during the execution, *postconditions* ϕ^{post} that constrain the state of the information space after successful execution, and *effects* ϕ^{eff} as conditions that hold in the world after successful execution. Defined with respect to ontologies $\mathcal{O}$, these are closed formulae specified in WSML [3].

So-called *shared variables* whose scope is the complete capability allow to specify the dependencies among the formulae. We replace this by a more precise model from the ASS model that explicitly defines in- and outputs [13]. The set of variables $IN = (i_1, \ldots, i_n)$ denotes all required inputs for invoking a Web service, respectively for instantiating a goal template. Their scope is the complete capability, and they occur as the only free variables in the $\phi^{pre}, \phi^{ass}, \phi^{post}, \phi^{\mathit{eff}}$ formulae. The set of universally quantified variables $OUT = (o_1, \ldots, o_m)$ denotes all outputs that are constrained by ϕ^{post}.

We further take on extensions from the ASS model that allow to more precisely describe the changes of the world performed by Web service executions. The extended signature $\Sigma_{\mathcal{A}}$ differentiates *static symbols* Σ_S that are not changed by the execution of a Web service, and *dynamic symbols* Σ_D that are changed by the execution. Besides, a capability carries non-functional properties $\mathcal{NFP}$ that include a natural language description and quality-of-service aspects. Definition 1 specifies the formal meaning of a capability in an Abstract State Space $\mathcal{A}$. Here, $s \models_{WSML} \phi$ expresses that the formula ϕ is satisfied under a Σ-interpretation over the universe $\mathcal{U}_{\mathcal{A}}$ that is assigned to a state $s \in \mathcal{A}$ with respect to the semantics of the WSML-variant wherein ϕ is specified.

Definition 1: A capability is a 9-tuple $\mathcal{C} = (\mathcal{O}, \Sigma_{\mathcal{A}}, IN, OUT, \phi^{pre}, \phi^{ass}, \phi^{post}, \phi^{\mathit{eff}}, \mathcal{NFP})$. *Let* $\tau = (s_0, \ldots, s_m)$ *be a sequence of state transitions in an Abstract State Space* $\mathcal{A}$.
The meaning of $\mathcal{C}$ *is that if* $s_0 \models_{WSML} \phi^{pre}$ *then* $s_m \models_{WSML} \phi^{post}$ *and* $s_m \models_{WSML} \phi^{\mathit{eff}}$ *if for all* $s \in \tau$ *holds* $s \models_{WSML} \phi^{ass}$.
If this holds, we say that τ *satisfies* $\mathcal{C}$, *denoted by* $\tau \models_{\mathcal{A}} \mathcal{C}$.

This essentially defines the meaning of a capability as an implication between the start- and the termination state of a sequence of state transitions. The overall functionality provided by a Web service W is the set of possible executions $\{\tau\}_W$. We say that W provides the functionality described by $\mathcal{C}_W$, denoted by $W \models_{\mathcal{A}} \mathcal{C}_W$, if for all $\tau \in \{\tau\}_W$ holds $\tau \models_{\mathcal{A}} \mathcal{C}_W$. The meaning of a capability $\mathcal{C}_{\mathcal{G}}$ that describes a goal $\mathcal{G}$ is analogous. Here, $\{\tau\}_{\mathcal{G}}$ is the set of sequences of state transitions that are solutions for $\mathcal{G}$ such that for all $\tau \in \{\tau\}_{\mathcal{G}}$ holds $\tau \models_{\mathcal{A}} \mathcal{C}_{\mathcal{G}}$. Capabilities can only be evaluated if there are concrete value assignments for the IN-variables. These are defined by goal instances as we explain the following.

B. Goal Instances

A goal instance formally describes a concrete client request. It is created at runtime by instantiating a goal template. In accordance to the Web service discovery framework envisioned for WSMO [12], the client therefore searches the repository for an appropriate goal template and then instantiates this with concrete input values. These values constitute the inputs for invoking a Web service to solve the goal instance.

The creation of a goal instance $GI(\mathcal{G})$ is achieved by defining an *input binding* β for the IN-variables in the capability $\mathcal{C}_{\mathcal{G}}$ of the chosen goal template $\mathcal{G}$. Formally, $\beta : \{i_1, \ldots, i_n\} \rightarrow \mathcal{U}_{\mathcal{A}}$ is a total function that assigns objects of the universe $\mathcal{U}_{\mathcal{A}}$ to each IN-variable defined in $\mathcal{C}_{\mathcal{G}}$. We therewith obtain an assignment of concrete values v for all required inputs, i.e. $\beta = \{i_1|v_1, \ldots, i_n|v_n\}$. This is considered to be valid for a capability $\mathcal{C}$ if the current state of the world s_c that β refers to satisfies the precondition, i.e. if $s_c, \beta \models_{WSML} \phi^{pre}$.

Given a valid input binding, we can determine the end-state of the particular solution for $GI(\mathcal{G})$ evaluating the semantics of $\mathcal{C}$ from Definition 1. We therefore instantiate the capability $\mathcal{C}_{\mathcal{G}}$ of the goal template by substituting all occurrence of an IN-variable by the value v assigned in β for all $i \in IN$ in each statement ϕ in $\mathcal{C}_{\mathcal{G}}$.

In consequence, it holds that the possible solutions for $GI(\mathcal{G})$ are a subset of those for $\mathcal{G}$, i.e. $\{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}}$. Because of this, we can realize a 2-phase discovery procedure: usable Web services for goal templates are determined at design time; discovery for goal instances is performed at runtime, whereby only those Web services need to be investigated that are usable for the corresponding goal template. This allows to achieve a better runtime efficiency. [22] discusses this in detail and specifies semantic matchmaking techniques for discovery on the goal template and the goal instance level.

C. WG Mediators

A WG mediator connects a goal template with a Web service and carries all information relevant for automated invocation and consumption of the Web service. Such a WG mediator is defined for each Web service that is usable for solving the goal template. Extending their definition from WSMO [16], we describe WG mediators by five elements:

(i) the *source* is a goal template $\mathcal{G}$
(ii) the *target* is a Web service W
(iii) it can use data and process level *mediation facilities*
(iv) it carries the *matching degree* between $\mathcal{G}$ and W and
(v) a *client interface* for invoking and consuming W in order to solve $\mathcal{G}$.

(i) and (ii) define that WG mediator is always a directed relation from a goal to a Web service description. (iii) states that mediation techniques can be utilized in order to handle

and resolve potentially occurring heterogeneities between the goal and the Web service.[1] (iv) defines the relevant information on the usability of the target Web service. This is denoted by the respective matching degree determined by discovery (see above) for directly usable Web services, or by `candidate` if the target Web service can be used to solve the source goal template as part of a Web service composition.

(v) defines the *client interface*. It is specified by ontologized Abstract State Machines (ASM), i.e. the WSMO description for Web service choreography interfaces [20]. This consists of (1) a *state signature* Ω that describes which information are used in in- and outgoing messages by defining modes for concepts of the domain ontology along with a grounding to a communication technology for information exchange [14], and (2) of *transition rules* TR that define guarded rules for state changes in the ASM. The client interface is constructed as the compatible counterpart of the choreography interface of the target Web service. It defines compatible modes in Ω along with a subset of the transition rules such that at least one common execution path exists between the two interfaces. Therewith, a client interface carries all information required for automated Web service usage (*cf* second requirement in Section II-A). We refer to [7] for more details on this.

D. Composite Goals

As the final element in our model, a composite goal is a goal template that extends the formal objective specification with a desired workflow that shall be sustained during the resolution of the goal. This goal template type is applied when the specification of a desired workflow is necessary to properly specify the client objective, which becomes especially relevant in the context of business process management [10].

A composite goal is defined as a tuple $G_{composite} = (\mathcal{C}, Orchestration)$ such that $\mathcal{C}$ is a capability that describes the objective to be achieved, and $Orchestration$ specifies the desired workflow that shall be sustained during the resolution of the goal. This workflow is a collection of subgoals along with control- and data flow among them. Its meaning is that each solution for $G_{composite}$ must sustain the specified sequence of states whereby intermediate states might be traversed between each desired state. In the ASS model, this means that $\tau \in \{\tau\}_{\mathcal{G}_{composite}}$ if $\tau = (s_0, \ldots, s_j, \ldots, s_m)$ and (s_0, s_j, s_m) is the desired workflow.

We formally describe the desired workflow as a WSMO orchestration [18]. This extends the ontologized ASMs used for choreography descriptions with constructs for specifying the interaction with aggregated goals and Web services. The ASMs are described by a state signature Ω and by a set of transition rules TR, which are formed from a more general set then those used for WSMO choreography. The central extension is the *perform* construct that allows to invoke aggregated goals and Web services on the level of performances, i.e. distinct prototypical executions of the aggregated resources. Each perform construct is accompanied by a *ppMediator* that connects performances and defines the bindings between their inputs and outputs.

In a nutshell, an orchestration in a composite goal consists of three central elements: (1) the guarding condition in a transition rule defines the control flow, (2) the perform construct specifies the particular subgoal to be performed, and (3) the data flow is defined by the in- and output variable bindings in ppMediators. We shall illustrate this in Section IV-A, and refer to [18] for details on the constructs and semantics.

Analogously to basic goal templates, a goal instance for a composite goal $GI(\mathcal{G}_{composite})$ is created by defining an input binding β for the IN-variables in the capability $\mathcal{C}$ of $\mathcal{G}_{composite}$ (see Section III-B). Apart from instantiating $\mathcal{C}$, this β also provides initialization of the orchestration ASM. The input values defined by the client are then subsequently forwarded to the subgoals and the Web services that are used for solving $GI(\mathcal{G}_{composite})$.

IV. Illustrative Example

This section discusses an example for demonstrating the goal descriptions as defined above. We consider a scenario from the travel domain, which is often used for illustration.

The client objective is to book a return flight and a hotel for a trip with the following workflow that shall be sustained during resolution of this goal. At first, available flights shall be found, then available hotels shall be found. Then, the client wants to personally select the particular flight and hotel to be booked. Finally, first the flight shall be booked and then the hotel. Figure 2 provides a graphical illustration of the desired workflow that we describe as a composite goal in the following. Thereafter, we discuss the usability of different Web services for solving this goal.

With respect to space limitations, we here only model the most interesting aspects of the first two subgoals (flight and hotel search), referring to [21] for a comprehensive discussion.

A. Goal Descriptions

Listing 1 shows the capabilities of the subgoals as defined in Section III-A. We here only model the pre- and postconditions. An assumption would be that the origin and destination of

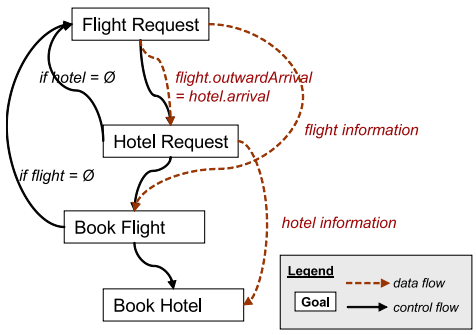

Fig. 2. Illustration of Desired Workflow

[1]The data level is concerned with heterogeneities between the ontologies used in the goal and Web service descriptions, and the process level is concerned with behavioral mismatches. [5] explains how these can help to establish successful Web service usage if this is not given a priori.

a flight actually are the airports associated with a city; an effect would be that the price for a booked flight or hotel is withdrawn from the client's bank account.

```
goal flightSearch capability
 inputVariables {?orig,?dest,?out,?in}
 outputVariables {?f}
 precondition definedBy
  ?orig memberOf city and ?dest memberOf city and
  ?out memberOf dateandtime and ?in memberOf dateandtime.
 postcondition definedBy
  ?f memberOf flight and
  ?f[ origin hasValue ?orig, destination hasValue ?dest,
      outwardDeparture hasValue ?od, outwardArrival hasValue ?oa,
      inwardDeparture hasValue ?id, inwardArrival hasValue ?ia] and
  after (?od, ?out).

goal hotelSearch capability
 inputVariables {?c,?a,?d}
 outputVariables {?h}
 precondition definedBy
  ?c memberOf city and
  ?a memberOf dateandtime and ?d memberOf dateandtime.
 postcondition definedBy
  ?h memberOf hotel and
  ?h[ city hasValue ?c, arrival hasValue ?a, departure hasValue ?d].
```

Listing 1. Capabilities of Sub Goals

Listing 2 shows the specification of the composite goal as introduced in Section III-D. The first part shows the goal capability similar to the ones above. The second part show the orchestration specification, and the ppMediators that bind the inputs and outputs of the orchestrated goals.

```
goal flightHotelGoal
capability
 inputVariables {?home,?venue,?leave,?return}
 outputVariables {?f, ?h}
 precondition definedBy
  ?home memberOf city and ?venue memberOf city and
  ?leave memberOf dateandtime and ?return memberOf dateandtime.
 postcondition definedBy
  ?f memberOf flight and
  ?f[ origin hasValue ?home, destination hasValue ?venue,
      outwardDeparture hasValue ?od, outwardArrival hasValue ?oa,
      returnDeparture hasValue ?rd, returnArrival hasValue ?ra] and
  after (?od, ?leave) and
  ?h memberOf hotel and
  ?h[ city hasValue ?venue, arrival hasValue ?oa,
      departure hasValue ?rd].
interface
 orchestration
  stateSignature
   shared { city, dateandtime }
   in { flight , hotel }
   controlled { cs1, cs2, ... }
  transitionRules
   forall {?cs} with (?cs[value hasValue cs1] memberOf control) do
     perform applyMediation ppFlightSearch_input
     perform subgoal1 achieveGoal FlightSearch
     update cs[value hasValue cs2]
   endForAll
   forall {?cs} with (?cs[value hasValue cs2] memberOf control) do
     perform applyMediation ppHotelSearch_input
     perform subgoal2 achieveGoal HotelSearch
     update cs[value hasValue cs3]
   endForAll
   [...]

ppMediator ppFlightSearch_input
 source parent
 target subgoal1
 definedBy ?orig = ?home and ?dest = ?venue and
           ?out = ?leave and ?in = ?return.
ppMediator ppFlightSearch_input
 source parent,subgoal1
 target subgoal2
 definedBy ?c = ?venue and ?a = ?oa and ?d = ?rd.
```

Listing 2. Composite Goal Description

In the orchestration, the state signature defines the concepts used in in- and outgoing messages: instances of *city* and *dateandtime* are sent and received, and *flight* and *hotel* are only received from the aggregated subgoals. The first transition rule is concerned with the flight search. At first, the ppMediator binds the inputs between from parent goal *flightHotelGoal* to the *flightSearch* goal. Variable bindings are defined via equality; this can be augmented with namespaces in the case of identical variable names. Then, the fight search subgoal shall be achieved. The second transition rule for the hotel search works analogously. To ensure that the hotel is available on the correct dates, the applied ppMediator binds the arrival and departures dates for the hotel search to the respective dates of the outward and return flights from the first subgoal.

Referring to [18] for details on the syntax and semantics, a main merit of this orchestration description is the concurrent handling of multiple goal instances at the same time. The $forall\{?cs\}$ guards define that the transition rule is executed for each goal instance whose resolution process is in the named control state. Besides, the editing and management of orchestration descriptions in supported by a 3-layer description language developed in the DIP project: UML Activity Diagrams serve as the graphical representation layer, the CASHEW layer handles the higher level workflow aspects, and the ASM layer illustrated here denotes the execution layer [24].

B. Relationship to Semantic Web Service Techniques

A central feature of our goal model is that the goal template types are orthogonal to the need for discovery (detection of directly usable Web services) or composition (combination of several Web services). Their usage is only dependent of the client objective: if a specific workflow is not desired, then the objective is described by a basic goal template. To demonstrate this, let us consider three Web services for the example:

- WS_1: a "trip booking Web service" where complete trips can be searched and booked (incl. flight, hotel, etc.)
- WS_2: a "flight booking Web service" where flights can be searched and booked (e.g. for a particular airline)
- WS_3: a "hotel booking Web service" where hotels can be searched and booked (e.g. `www.hotels.com`).

Here, either WS_1 or a composition of WS_2 and WS_3 can be used to solve the composite goal specified above. WS_1 can provide a solution for the composite goal when considering the capability level. If the client interface that is constructed with respect to the choreography interface of WS_1 sustains the workflow specified in orchestration of *flightHotelGoal*, then all relevant aspects of the client objective are satisfied.

The composition $WS_2 \circ WS_3$ needs to define an interleaved consumption of the Web services in order to sustain the desired workflow. At first, the search facility of WS_2 is consumed by the flight search subgoal, followed by the search facility of WS_3 for the hotel search; then, the flight is booked via WS_2, and finally the hotel via WS_3. If such a composition can be constructed, then it can be be used to solve *flightHotelGoal*.

V. Conclusions and Related Work

This paper has presented a description model for goals as the client side element for enabling problem-oriented usage of Web service. Based on the experiences gathered in the DIP project, we have extended the initial WSMO goal model by:

1) the differentiation of goal templates and goal instances that allows to perform expensive discovery and composition operations at design time and support goal formulation by clients via graphical user interfaces
2) the use of WG mediators that connect goal templates with usable Web services, carry all information for automated invocation and consumption in form of client interfaces, and utilize mediation facilities to handle potentially occurring heterogeneities
3) composite goals for specifying restrictions on the workflow for solving a goal.

Regarding related work, WSMO is the only framework for Semantic Web services that encompasses goals as a top level notion. OWL-S merely provides a meta-model for semantically describing Web services. Respective works for discovery and composition define client requests in form of queries (e.g. [11], [19]). This is a more technical perspective which we do not consider as a sophisticated model for supporting problem-oriented Web service usage. While WSMO initially does not distinguish goal templates and goal instances, related system implementations such as IRS [4] or SWF [23] have successfully deployed this, therewith achieving a better scalability and providing graphical support for goal formulation by clients.

The conception of goals as formalized client requests has been inspired by approaches for automated problem solving from different AI disciplines (esp. cognitive architectures [17] and agent BDI architectures [2]). In contrast to [15], we describe the requested functionality in goal templates by preconditions and effects. The reason is that in service-oriented architectures usually the current state of the world is not explicated or not accessible to the interaction partners. Regarding composite goals, related efforts have been presented in the context of Web service composition (e.g. [1], [6]). Therein, so-called composition goals extend the requested functionality by workflow or process restrictions on acceptable compositions. In our model, the goal template type is orthogonal to the need for discovery or composition.

This paper has presented the conceptual model and definition of our goal model. Ongoing efforts that will be continued in the future are the integration into the WSMO specification as well as the further development of software systems for goal-driven Web service usage along with graphical support for goal creation and administration.

Acknowledgment

This material is based upon works supported by the EU under the projects DIP (FP6 - 507483) and SUPER (FP6 - 026850). We like to thank Laurent Henocque, Jens Lemcke, and John Domingue for extensive discussions in the DIP project, and the members of the WSMO working group (`www.wsmo.org`) for feedback on the present work.

References

[1] P. Albert, L. Henocque, and M. Kleiner. Configuration-Based Workflow Composition. In *Proc. of 3rd International Conference on Web Services (ICWS-05), Orlando, Florida*, 2005.

[2] M. E. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA (USA), 1987.

[3] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Deliverable D16.1 final draft 05 Oct 2005, WSML Working Group, 2005. online: http://www.wsmo.org/TR/d16/d16.1/v0.2/.

[4] L. Cabral, J. Domingue, S. Galizia, A. Gugliotta, B. Norton, V. Tanasescu, and C. Pedrinaci. IRS-III – A Broker for Semantic Web Services based Applications. In *In Proc. of the 5th International Semantic Web Conference (ISWC 2006), Athens(GA), USA*, 2006.

[5] E. Cimpian, A. Mocan, and M. Stollberg. Mediation Enabled SemanticWeb Services Usage. In *Proc. of the 1st Asian Semantic Web Conference (ASWC 2006), Beijing, China*, 2006.

[6] U. Dal Lago, M. Pistore, and P. Traverso. Planning with a Language for Extended Goals. In *Proc. of 18th National Conf. on Artificial Intelligence (AAAI'02)*, 2002.

[7] J. Domingue, S. Galizia, and L. Cabral. The Choreography Model for IRS-III. In *Proc. of the 39th Hawaiian International Conference On System Sciences (HICSS-39), Kauai Island, Hawaii*, 2006.

[8] T. Erl. *Service-Oriented Architecture (SOA). Concepts, Technology, and Design*. Prentice Hall PTR, 2005.

[9] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domigue. *Enabling Semantic Web Services. The Web Service Modeling Ontology*. Springer, 2006.

[10] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel. Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In *Proc. of the IEEE ICEBE 2005, Beijing, China,*, 2005.

[11] D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding Semantic Matching of Stateless Services. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI'2006)*, 2006.

[12] U. Keller, R. Lara, H. Lausen, and D. Fensel. Semantic Web Service Discovery in the WSMO Framework. In J. Cardoses, editor, *Semantic Web: Theory, Tools and Applications*. Idea Publishing Group, 2006.

[13] U. Keller, H. Lausen, and M. Stollberg. On the Semantics of Funtional Descriptions of Web Services. In *Proc. of the 3rd European Semantic Web Conference (ESWC 2006), Montenegro*, 2006.

[14] J. Kopecký, D. Roman, M. Moran, and D. Fensel. Semantic Web Services Grounding. In *Proc. of the International Conference on Internet and Web Applications and Services (ICIW'06), Guadeloupe, French Caribbean*, 2006.

[15] R. Lara, M. A. Corella, and P. Castells. A Flexible Model for Web Service Discovery. In *Proc. of the 1st International Workshop on Semantic Matchmaking and Resource Retrieval: Issues and Perspectives, Seoul, South Korea*, 2006.

[16] A. Mocan, E. Cimpian, and M. Stollberg (eds.). WSMO Mediators. Deliverable D29, 2005. online: http://www.wsmo.org/TR/d29/.

[17] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA (USA), 1990.

[18] B. Norton. Dataflow for Orchestration in WSMO. Deliverable D15.1, 2006. online: http://www.wsmo.org/TR/d15/d15.1/.

[19] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *Proc. of the First International Semantic Web Conference, Sardinia, Italy*, 2002.

[20] D. Roman and J. Scicluna. Ontology-based Choreography of WSMO Services. Deliverable D14, 2006. online: http://www.wsmo.org/TR/d14/.

[21] M. Stollberg and M. Hepp. Goal Description Ontology. Deliverable D3.10, DIP, 2006.

[22] M. Stollberg and U. Keller. Semantic Web Service Discovery: Matchmaking for Goal Templates and Goal Instances on Rich Functional Descriptions. Technical Report DERI-2006-10-20, DERI, 2006.

[23] M. Stollberg, D. Roman, I. Toma, U. Keller, R. Herzog, P. Zugmann, and D. Fensel. Semantic Web Fred – Automated Goal Resolution on the Semantic Web. In *Proc. of the 38th Hawaii International Conference on System Science*, 2005.

[24] M. Stollberg et al. DIP Interface Description Ontology. Specification, DIP, 2005. online: http://dip.semanticweb.org/documents/DIO-Annex-to-D3.4-and-D3.5.pdf.