Goal Refinement for Automated Service Discovery

Tomas Olsson, May Yee Chong, Björn Bjurling
Swedish Institute of Computer Science
164 29 Kista, Sweden
Email: {tol, may, bgb}@sics.se

Börje Ohlman
Ericsson Research AB
Färögatan 6, 164 80 Kista, Sweden
Email: borje.ohlman@ericsson.com

***Abstract*—An important prerequisite for service composition is a versatile and efficient service discovery mechanism. The trends in service computing currently point toward a veritable explosion in the number of services that are or will become available as components in compositions of new services—Cloud Computing and the 'Internet of Things' are just two examples of such new trends. We hypothesize that it will become critical to be able to filter the discovered services with respect to pre- and postconditions, as well as other semantic aspects such as relevance. We have studied the use of *goals* as a means for describing semantic aspects of services (e.g., their effects). By using goals, services can be described on any arbitrary and useful level of abstraction. By a goal refinement algorithm, goals can be used not only for describing services, but also for improving the performance of service discovery. In this paper, we describe the goal refinement algorithm and our approach to incorporating it into our service discovery machinery.**

***Keywords*-Service Discovery; Service Composition; Semantic Web Service; Goal Refinement.**

I. INTRODUCTION

One major trend of today is to transition old and new software into a service-oriented environment for reusing and sharing services across organizational boundaries. Key enabling technologies for this transition are web service technologies such as WSDL [1] and SOAP [2]. As the number of services placed online increases, methods for simplified and intelligent service discovery have become an important area of research [3][4]. Service discovery is an essential prerequisite of service composition, and it is paramount that the service discovery mechanism can determine whether a service is relevant to the requirements in a service composition specification.

One avenue of approach to finding a solution to the discovery and composition problem is to use semantic web services where the services are described using ontologies [5][6]. A step toward this is the Web Service Modeling Ontology (WSMO) [5] and the corresponding Web Service Modeling eXecution (WSMX) framework [7]. Another similar ontology for modeling web services is OWL-S [6]. In particular interest for our work is the Web Service Modeling Language (WSML) based on WSMO for semantically describing ontologies, goals, and web services.

The idea of using goals, as, e.g., in WSMO, is appealing in that goals allow us to express the desired outcome of a service, i.e., as a description of the resulting system state. Note that goals thus can be formulated without reference to APIs of specific services, where merely a desired behavior can be expressed. Following Kavakli et al. [8], we say informally that a goal is a "*...a desired condition potentially attained at the end of an action (or a process)*".

This paper is a continuation of our previous work [9][10] where we presented an approach for facilitating the setting-up and management of new multi-organizational services using goals. We have in this paper also used ideas and concepts from the WSMO conceptual framework. In contrast to WSMO and OWLS-S, we propose a bottom-up approach where semantic annotations are added to the operations of each web service along the lines suggested by SAWSDL [11], and similarly to [12]. An operation annotation is a description of input/output variables, assumptions, and effects. The assumptions and effects are also expressed as goals.

We define a goal formally as a logical expression in the WSML language [13]. Note that the notion of a goal in WSML is more complex. Our aim is to let the discovery mechanism search for combinations of services that can fulfill a user-provided goal. For that purpose, we have developed a goal-refinement algorithm and incorporated it into our service discovery algorithm. Goal refinement breaks down a goal into more specific goals with which we can obtain improved matching results. Goal refinement also determines the set of operations needed for fulfilling the goal.

The contribution of this work is the combination of: (1) a goal-driven discovery mechanism for finding services, and (2) a refinement algorithm that decomposes a goal to subgoals and test whether a goal is fulfilled, and (3) a bottom up approach for semantic web service modeling with annotation at operation level. These three components make a powerful combination by extending discovery of web services by also telling exactly which operations of the discovered services that should be used in a composition.

This report is structured as follows. In Section II, we present related work. Section III, describes WSML. Section IV presents the goal-driven automated service composition framework including semantic modeling of web services. Section V explains the refinement process and provides an example. The use of refinement in the service discovery process is explained in Section VI. Section VII concludes with a summary and a discussion of the current limitations of our approach.

II. RELATED WORK

The Universal Description, Discovery, and Integration protocol (UDDI) [14] is an industrial initiative for enabling the

 ISBN: 978-1-61208-152-6

discovery and publishing of web services in a distributed way. The keyword based discovery mechanism in UDDI does not support discovery based on semantic descriptions of a service. It is thus not possible to locate a service based on what problem it solves. As a complement to UDDI, semantic matching between web services was proposed by Paolucci et al. [15]. The authors describes a model for creating semantic service profiles in DAML-S.

The Web Service Modeling Ontology (WSMO) [5] is a conceptual framework for modeling discovery and composition of web services. In WSMO, web services and goals are modeled using semantic information in the form of preconditions, postconditions, assumptions, and effects. The interaction protocol of services are described by choreographies. For service discovery, goals are matched against web service descriptions. A similar top-down approach is OWL-S [6] where a service is modeled using four ontologies: an upper Service ontology that links a *Service Profile* ontology, used for service discovery, a *Service Model* ontology, used for modeling client-service interaction and lastly a *Service Grounding* ontology, used for binding the service description into corresponding WSDL entities. A difference between OWLS-S and WSMO is that WSMO makes a distinction between goals and web services while OWL-S uses the service profile for both.

The approach of modeling services and then grounding them in WSDL files, can result in fairly large service descriptions. WSMO-lite [12] addresses this limitations of WSMO by providing a bottom-up approach where operations of a WSDL file are semantically annotated using SAWSDL [11]. This makes the service models smaller in that the service choreography automatically can be inferred from the annotations. In [16], the authors extends SAWSDL with the definition of pre- and postconditions using SPARQL query language and OWL-S Schema. The SPARQL query language is also used to formulate goals by users.

Another approach is presented in [17] where the authors introduce a goal template to precompute potential web services and goal instances that correspond to concrete client requests. Both goals and web services are modeled semantically using first-order logic as state transitions from an initial state. A similar work was presented in [18] but without goal templates. Goals are considered only as final states, as in this paper. Neither of these two papers consider annotations on operations.

The goal-based approach presented by Bandara et al. [19] uses goal elaboration based on the KAOS method [20] combined with abductive reasoning to infer the mechanisms by which a given system can achieve a particular goal. This provides for partial automation of the, possibly inconsistent and incomplete, manual KAOS approach [20]. The Goal-Based Service Framework for dynamic service discovery and composition is described in [21]. The framework allows modeling and inference between high-level goals and other parts of the system, such as a domain model, services, clients, and high-level tasks.

Table I
MESSAGING SERVICE ONTOLOGY

concept sendEntity
addresses ofType (1 *) _IRI
sender ofType (0 1) _string
messageIdentifier ofType _string
deliveryResult ofType (0 *) DeliveryInformation
dateTime ofType (0 1) _dateTime

concept SMSEntity **subConceptOf** sendEntity
message ofType _string

concept MMessageEntity **subConceptOf** sendEntity
subject ofType (0 1) _string
priority ofType (0 1) MessagePriority

III. THE WEB SERVICE MODELING LANGUAGE

WSML is defined with syntax and semantics closely following WSMO [13]. The two basic components of the WSML syntax are the *conceptual syntax* and *logical expression syntax*. The conceptual syntax is used for modeling Ontologies, Web services, Goals, and Mediators, while the logical expression syntax is used for making logical inference over these definitions. In our work, we use only a subset of the syntax: ontologies and logical expressions.

An ontology in WSML consists of concepts, their attributes and the relations between instances of the concepts. An example of a ontology for describing Messaging services is shown in Table I. A generic Message concept is encapsulated in the *sendEntity* item with the attributes, cardinality, and their value type. A more concrete instance of this messaging concept is the *SMSEntity* and the *MMessageEntity* entities, both of which inherit the attributes of their superconcept in addition to specific attributes.

The vocabulary of the logical expression syntax consists of *identifiers*, *object constructors*, *function symbols*, *datatype wrappers*, *data values* that includes all string, integer and decimal values, *anonymous identifiers*, *relation identifiers* and *variable identifiers* of the form ?alphanum*. In the proposed work, we allow a subset of WSML's logical connectives: *and*, *implies*, *impliedBy*, *equivalent*, and the auxiliary symbols: '(', ')', '[', ']', ',', '=', '!=', *memberOf*, *hasValue*, *subConceptOf*, *ofType*, and *impliesType*. For instance, "$?msg$ **memberOf** $MMessageEntity$ **implies** $?msg$ **memberOf** $sendEntity$" (where $?msg$ is a variable identifier) is a valid logical expression in the Ontology in Table I. It says that an instance of the concept *MMessageEntity* is also an instance of the concept *sendEntity*.

IV. GOAL-DRIVEN AUTOMATED SERVICE COMPOSITION

The long term vision of this work is to provide a framework where new services can easily be created. A user should be able to specify high-level business goals, in terms of QoS parameters, Key Performance Indicators and, of course, specific functionality. Then, the framework would automatically refine it into a running, self-managing service. In this context, we define a goal g to be a WSML-expression describing a desired

 ISBN: 978-1-61208-152-6

state of a system. Thus the framework should create a service that will, when run, achieve the goal.

We have implemented a prototype of the framework consisting of three components: *service discovery engine*, *service creation engine*, and *service execution component*. Given a high-level goal, the service discovery engine tries to find a set of web service operations that fulfills the goal. The service creation engine uses the service discovery engine recursively to find all required dependencies and then composes the operations into an executable service. The service execution component then runs the composite service. In this paper, we present details of the service discovery implementation.

A. Semantic Modeling

The framework uses a bottom-up approach to semantic modeling where we annotate WSDL files with semantic information per operation. Thus we represent a web service as a set of semantically annotated operations. Each operation of a web service is modeled as a set of *input variables* I, a set of *output variables* O and a set of *state transitions* T. A transition consists of an assumption a and an effect e, where a and e are WSML-expressions. The assumption describes the state in which the operation can be applied and the effect describes the resulting state from applying the operation.

By modeling on the operation level, we can automatically generate the interaction protocol from the annotations and thus we do not need the choreography of WSMO or the Service Model of OWL-S. In addition, a goal can be much smaller than in WSMO since there is no correspondence between a goal and a web service. Instead, goals are matched to operation effects, and the assumptions can then in turn be seen as additional goals that infer dependencies between operations.

B. Semantic Matching

In line with previous work in semantic matching of web services, we define a set of categories of matching between a goal g and an operation effect e: *plugin*, *subsume*, *exact* and *no match* [15][22][7]. We use $match(g,\ e)$ to denote the type of matching, such that $match(g,\ e) = Subsume$, $match(g,\ e) = Plugin$, $match(g,\ e) = Exact$ and $match(g,\ e) = NoMatch$ denotes the corresponding matches. In a subsume match, the g is more generic than e. For instance, g subsumes e whenever $g = ?user\ \mathbf{memberOf}\ User$ while $e = ?user\ \mathbf{memberOf}\ RootUser$ given that $RootUser\ \mathbf{subConceptOf}\ User$. In case of a plugin match, the situation is the opposite; the e is more generic than g. For instance, if $g = ?user\ \mathbf{memberOf}\ User\ \mathbf{and}\ ?user\ \mathbf{memberOf}\ Police$ while $e = ?user\ \mathbf{memberOf}\ User$, then e subsumes g and thus, it is a plugin match. In case that there are both a plugin match and a subsume match for two expressions, it is an exact match. If there is neither subsume nor plugin matches, then it is said to be a *no match*.

For implementing the matching operator, we use the algorithm implemented for checking query containment in the WSML2Reasoner framework [23]. As inference engine, we use the "IRIS well grounded" engine.

V. Goal Refinement

The discovery process starts off by the specification of a goal for a new service and proceeds by searching for sets of operations that can implement the new service by matching the goal with the effect of the operations in the annotated web services. This process is implemented using our *goal refinement algorithm*. The rest of the paper describes the implementation of goal refinement and the incorporation of it into the service discovery process.

Goal refinement requires domain knowledge of services and their components. This is described semantically by using one or more ontologies written in WSML. Algorithm 1 describes the refinement process of a goal and outputs a set of goals where each item is a refined expression of the original.

Given an input goal, the refinement algorithm decomposes the input expression into a set of atomic expressions before performing refinement on each atomic goal. Goal Decomposition, the first step of goal refinement, is the simplification of complex goals by the removal of logical connectives. This is performed using the Lloyd Topor Normalizer in the WSML2Reasoner framework [23].

Refinement of each atomic element is performed using the process described in Algorithm 1. In each atomic expression, the concept is identified (3.2) and its list of subconcepts is retrieved from knowledge represented by the ontology (3.3).

Algorithm 1 Simple Goal Refinement

```
1:  procedure RefineSimpleGoal(g - a goal)
2:      G' ← decompose(g)                          ▷ - {3.1}
3:      SG ← ∅                                     ▷ - a set of subgoals
4:      for all g' ∈ G' do
5:          c ← findConcept(g')                    ▷ - {3.2}
6:          SC ← listSubConcepts(c)                ▷ - {3.3}
7:          if SC is not empty then
8:              for all sc ∈ SC do
9:                  g'' ← g'
10:                 replace c in g'' with sc
11:                 insert g'' into SG
12:             end for
13:         end if
14:     end for
15:     return SG
16: end procedure
```

We illustrate the algorithm by refining a simple goal. The ontology in Table I provides the context for the goal, and a messaging concept is represented using an entity *sendEntity* which consists of two smaller entities, *SMSEntity* and *MMSEntity*. The former represents a simple text message and the latter a multimedia message. Assume that we are given a goal requesting a generic messaging service. This could be the case when the requester doesn't know or doesn't care how a message should be forwarded. Since the goal is generic we may assume that there is no service available matching the

Table II
INPUT AND REFINED OUTPUT EXPRESSIONS

Input Expression:
?x memberOf msgOnt#sendEntity.
Output Expression Set:
?x memberOf msgOnt#MMSEntity.
?x memberOf msgOnt#SMSEntity.

goal, and that we thus need to refine it into more specific goals which in turn may be matched by existing services.

The goal expresses a requirement for a generic messaging service, described as the input expression in Table II. The generic messaging ontology given above, with its two subconcepts is used for refining the goal. The goal is refined into two subgoals containing the two subconcepts. Table II shows the original (input) and refined (output) expressions from the algorithm. Each subgoal may be satisfied by different service providers: a text messaging service provider and a multimedia messaging service provider. Provision of both services will fulfill each of the subgoals and in turn the original goal.

This example illustrated that a single provider may not possess all the resources necessary to fulfil a goal and that refinement into more specific subgoals may help aggregating providers whose services may fully match the given goal.

In the discovery process (see Section VI), refinement is performed when a plugin or subsume match has been obtained during the matching of a goal and a web service operation. In the case of a plugin match between a service provider and a goal consumer, the goal is more specific than the web service. A refinement of the web service is then performed and each refined web service is matched with the goal. With a subsume match, the web service is more specific than the goal, in which case the goal is refined and for each refined subgoal a matching with the web service is performed.

VI. GOAL REFINEMENT IN SERVICE DISCOVERY

Service discovery is achieved by matching a goal provided by a user against the operations of all web services using goal refinement. By integrating goal refinement into the discovery process, we improve match results by being able to combine subsume and plugin type matches. In addition, we extract common and differing parts of expressions when matching a goal to a web service operation. This information is used to modify the goal if necessary and in further matching processes.

We produce three sets of expressions in this extraction process. The common expressions, *CO*, represent the resources required by the consumer to be fulfilled by the provider. Goals required by consumer that cannot be fulfilled by the provider, *GO*, form a new goal and is used to match against services by other providers. Providers who offer resources that are not needed by the consumer, *WO*, can be offered to other consumers, thus optimizing resource usage. We demonstrate in Algorithm 2 the process of comparing a goal and a web service with the refinement and extraction processes introduced.

By comparing all the operations in a web service against a goal, as implemented in Algorithm 2, we select the operation that best fulfills a goal. This is done by selecting the operation that has the most in common with the goal, while keeping the unfulfilled goal size to a minimum and using as much of the web service operation as possible.

For each web service operation and goal, we use Algorithm 2 to obtain the matching result, rank these results and select the most suitable web service operation. The goals left unfulfilled by the selected web service operation are formulated as a new goal which is to be matched against other web services in the next iteration of the refinement. In the current version of our implementation, only one operation is selected from each matching web service.

Algorithm 2 Refinement in Matching

```
1:  procedure (g - a goal, e - an effect)
2:      C ← ∅                                  ▷ : In both goal and effect.
3:      GO ← ∅                                 ▷ :Found only in goal.
4:      WO ← ∅                                 ▷ :Found only in effect.
5:      G' ← decompose(g); E' ← decompose(e)
6:      for all g' ∈ G' do
7:          for all e' ∈ E' do
8:              type ← match(g', e')
9:              if type = Exact then
10:                 insert g' into C; remove e' from E'
11:                 matched ← true
12:             else if type = Subsume then
13:                 Handle subsume             ▷ :See Algorithm 3.
14:                 remove e' from E'; matched ← true
15:             else if type = Plugin then
16:                 Handle Plugin              ▷ :See Algorithm 4.
17:                 remove e' from E'; matched ← true
18:             end if
19:             if matched = true then
20:                 break loop
21:             end if
22:         end for
23:         if matched = false then
24:             insert g' into GO
25:         end if
26:     end for
27:     WO ← E'
28:     return ⟨C, GO, WO⟩
29: end procedure
```

Algorithm 3 Handle Subsume Matches

```
1:  SG ← refine(g)                             ▷ : Refine Goal into Subgoals.
2:  for all sg ∈ SG do
3:      subtype ← match(sg, e)
4:      if subtype = NoMatch then
5:          insert sg into GO
6:      else if subtype = Exact then
7:          insert sg into C
8:      else
9:          Compare attributes between sg and e
10:         Return common, onlyGoal and onlyWS attributes
11:     end if
12: end for
```

Algorithm 4 Handle Plugin Matches

```
1: E' ← refine(e)                          ▷ : Refine web service effects.
2: for all e' ∈ E' do
3:     subtype ← match(g, e')
4:     if subtype = NoMatch then
5:         insert e' into WO
6:     else if subtype = Exact then
7:         insert e' into C
8:     else
9:         Compare attributes between g and e'
10:        Return common, onlyGoal and onlyWS attributes
11:    end if
12: end for
```

A. An Example

We provide an example on service discovery performed together with refinement and extraction methods. Our user-provided goal is to create a service that is able to deliver a message to a registered user. A number of ontologies are used to describe this goal and a number of web service providers have services available for the automated composition process. In our tests, we have modeled an Information service, SMS service and MMS service among others. Table III shows the user-provided goal, $G0$ and some operations ($OP1$, $OP2$, and $OP3$) in available web services. Table IV shows the results from attempting to match $G0$ with each of the web services.

As implemented in Algorithm 2, the goal is first decomposed into a set of *atomic* (with respect to the refinement algorithm) goals and each component is compared against the web service operations. The original goal expression, $G0$ is compared to the first service (information service) and obtains the best match against the operation *createUser (OP1)*. The *createUser* web service operation partially fulfills the goal. The results obtained with the matching of each atomic expression are either exact matches or no match. In this step, no plugin or subsume matches have been obtained to warrant refinement. *F1* of the goal is fulfilled while the remaining unfulfilled goal is formulated into a new goal $G1$ and used in subsequent matches with other web services.

$G1$ is matched with the operations in the next web service, an SMS web service. *F2* shows the goals that can be fulfilled by the *sendSMS (OP2)* operation in the web service and the remaining unfulfillable goals $G2$ are used in subsequent matches. In this comparison, the atomic expression in $G1$, *?msg[msgOnt#deliveryResult hasValue ?x] memberOf msgOnt#sendEntity*, is a subsume match to the atomic expression of *?msg[msgOnt#deliveryResult hasValue ?x] memberOf msgOnt#SMSEntity*. Refinement of the $G1$ yields two subgoals: *?msg[msgOnt#deliveryResult hasValue ?x] memberOf msgOnt#SMSEntity* and *?msg[msgOnt#deliveryResult hasValue ?x] memberOf msgOnt#MMSEntity*. The former is an exact match and the latter, $G2$, is added to the list of goals that need to be fulfilled by other providers.

$G2$ is matched against the operations in the next web service, an MMS web service. *F3* shows the goals that can be fulfilled by the *sendMMS* operation in the web service and the remaining *G3* remains unfulfilled.

Table III
COMPARING GOAL AND WEB SERVICE EXPRESSIONS

Goal Expression, *G0* :
?user[ug#name hasValue "Tomas Olsson",
ug#mobilePhone hasValue ?mp
] memberOf ug#User and
my#isRegisteredByInfoService(?user) and
?msg[msgOnt#deliveryResult hasValue ?x
] memberOf msgOnt#sendEntity
and msgOnt#forDelivery(?msg).

Create User Operation from Information Service, *0P1*:
?newUser[ug#name hasValue ?userName
] memberOf ug#User and
my#isRegisteredByInfoService(?newUser).

Send SMS Operation from SMS Service, *0P2*:
?x[msgOnt#address hasValue ?address
] memberOf msgOnt#DeliveryInformation and
?msg[msgOnt#deliveryResult hasValue ?x
] memberOf msgOnt#SMSEntity and
msgOnt#forDelivery(?msg).

Send MMS Operation from MMS Service, *0P3*:
?x[msgOnt#address hasValue ?address
] memberOf msgOnt#DeliveryInformation and
?msg[msgOnt#deliveryResult hasValue ?x
] memberOf msgOnt#MMSEntity and
msgOnt#forDelivery(?msg).

VII. CONCLUSION AND FUTURE WORK

In this paper, we have described a goal-driven service discovery mechanism and an approach to annotating operations in semantic web services. We have described how our goal refinement algorithm can be used to refine a user provided goal into subgoals with which we can perform a fine-grained service matching in the service discovery process. Moreover, we have illustrated how non-matched subgoals can be extracted to form new goals for subsequent matching.

Our discovery process has not been optimised to reduce the time required to search for the best matching web service to the goal. Currently, the larger the number of web services in the repository, the more time is required to complete a matching. The complexity of the algorithms needs further investigation, and the algorithms should be improved accordingly. In our implementation, the order in which web services are matched affects the results. The resulting composition may be improved (e.g., with respect to the number of implementing services) by investigating methods for finding optimal orderings of the services prior to matching.

Using goals for modeling and finding services seems to be a promising approach. By specifying high-level goals it is possible to manage services at macro level and derive management of specific services at micro level using goal-refinement. In this work, we have used implicit goal-decomposition and goal-composition rules. However, to make the approach more flexible we need an explicit goal-translation modeling language, where a domain expert can explicitly model what high-level goals can be translated into what low-level goals.

 ISBN: 978-1-61208-152-6

Table IV
SERVICE DISCOVERY RESULTS

Results of Comparison Against Information Service

Goal Fulfilled, *F1*:
?user[ug#name hasValue "Tomas Olsson"] memberOf ug#User and my#isRegisteredByInfoService(?user).

Remaining Goal, *G1*:
?user[ug#mobilePhone hasValue ?mp] memberOf ug#User and ?msg[msgOnt#deliveryResult hasValue ?x] memberOf msgOnt#sendEntity and msgOnt#forDelivery(?msg).

Best Operation Match:
http://www.sics.se/gops/webservices/InfoService/createUser

Results of Comparison Against SMS Service

Goal Fulfilled, *F2*:
?msg[msgOnt#deliveryResult hasValue ?x] memberOf msgOnt#SMSEntity and msgOnt#forDelivery(?msg).

Remaining Goal, *G2*:
?user[ug#mobilePhone" hasValue ?mp] memberOf ug#User and ?msg[msgOnt#deliveryResult hasValue ?x] memberOf msgOnt#MMSEntity.

Best Operation Match:
http://www.sics.se/gops/webservices/SMSService/sendSMS

Results of Comparison Against MMS Service

Goal Fulfilled, *F3*:
?msg[msgOnt#deliveryResult" hasValue ?x] memberOf msgOnt#MMSEntity".

Remaining Goal, *G3*:
?user[ug#mobilePhone hasValue ?mp] memberOf ug#User.

Best Operation Match:
http://www.sics.se/gops/webservices/MMSService/sendMMS

Final Results

Final Set of Operations:
http://www.sics.se/gops/webservices/InfoService/createUser
http://www.sics.se/gops/webservices/SMSService/sendSMS
http://www.sics.se/gops/webservices/MMSService/sendMMS

Goal not fulfillable by any service:
?user[ug#mobilePhone hasValue ?mp] memberOf ug#User.

ACKNOWLEDGEMENT

The present work was funded by the GOPS project. GOPS is sponsored by Vinnova.

REFERENCES

[1] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web services description language (wsdl) 1.1," World Wide Web Consortium, W3C Note 15, March 2001.

[2] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon, "Soap version 1.2 part 1: Messaging framework (second edition)," World Wide Web Consortium, W3C Recommendation 27, April 2007.

[3] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Web service discovery mechanisms: Looking for a needle in a haystack?" in *International Workshop on Web Engineering*, August 2004.

[4] N. Steinmetz, H. Lausen, and M. Brunner, "Web service search on large scale," in *Proceedings of the 7th International Joint Conference on Service-Oriented Computing*, ser. ICSOC-ServiceWave '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 437–444.

[5] D. Roman and H. Lausen, "D2v1.3. web service modeling ontology (wsmo) - wsmo final draft 21 october 2006," [Online]. Available: http://www.wsmo.org/TR/d2/v1.3/ 29-06-2011.

[6] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "Owl-s: Semantic markup for web services," [Online]. Available: http://www.ai.sri.com/daml/services/owl-s/1.2/overview/ 29-06-2011.

[7] U. Keller, R. Lara, H. Lausen, A. Polleres, L. Predoiu, and I. Toma, "D10v0.2 semantic web service discovery wsmx working draft," [Online]. Available: http://www.wsmo.org/TR/d10/v0.2/ 29-06-2011.

[8] E. Kavakli and P. Loucopoulos, *Information Modelling Methods and Methodologies*, 2005, ch. Goal Modelling in Requirements Engineering: Analysis and Critique of Current Methods, pp. 102–124.

[9] Å. Berglund, B. Bjurling, R. Dantas, S. Engberg, P. Giambiagi, and B. Ohlman, "Towards goal-based autonomic networking," in *Third International Workshop on Distributed Autonomous Network Management Systems*, Nov 2008.

[10] M. Y. Chong, B. Bjurling, R. Dantas, C. Kamienski, and B. Ohlman, "Goal-based service creation using autonomic entities," in *MACE '09: Proceedings of the 4th IEEE International Workshop on Modelling Autonomic Communications Environments*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 29–43.

[11] J. Farrell and H. Lausen, "Semantic annotations for wsdl and xml schema," World Wide Web Consortium, W3C Recommendation 28, August 2007. [Online]. Available: http://www.w3.org/TR/sawsdl/

[12] T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel, "Wsmo-lite annotations for web services," in *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ser. ESWC'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 674–689.

[13] N. Steinmetz and I. Toma, "D16.1v1.0 wsml language reference - wsml final draft 2008-08-08," [Online]. Available: http://www.wsmo.org/TR/d16/d16.1/v1.0/ 29-06-2011.

[14] "Uddi," http://uddi.xml.org/ 29-06-2011.

[15] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," in *The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002. Proceedings*, 2002, pp. 333+.

[16] K. Iqbal, M. L. Sbodio, V. Peristeras, and G. Giuliani, "Semantic service discovery using sawsdl and sparql," in *Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 205–212.

[17] M. Stollberg, U. Keller, H. Lausen, and S. Heymans, "Two-phase web service discovery based on rich functional descriptions," in *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, ser. ESWC '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 99–113.

[18] R. Lara, "Two-phased web service discovery," in *AI-Driven Technologies for Services-Oriented Computing Workshop at AAAI-06*, Boston, USA, 2006.

[19] A. K. Bandara, E. Lupu, J. D. Moffett, and A. Russo, "A Goal-based Approach to Policy Refinement," in 5^{th} *IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, 2004, pp. 229–239.

[20] R. Darimont and A. van Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration," in *4th ACM SIGSOFT Symposium on Foundations of Software Engineering*, vol. 21, no. 6. ACM Press, Nov 1996, pp. 179–190.

[21] L. O. B. da Silva Santos, G. Guizzardi, L. F. Pires, and M. van Sinderen, "From user goals to service discovery and composition," in *Advances in Conceptual Modeling - Challenging Perspectives*, ser. Lecture Notes in Computer Science. Berlin: Springer Verlag, November 2009, vol. 5833, pp. 265–274.

[22] U. Keller, R. Lara, A. Polleres, I. Toma, M. Kifer, and D. Fensel, "WSMO Web Service Discovery," DERI, Tech. Rep. D5.1v0.1, November 2004.

[23] S. Grimm, U. Keller, H. Lausen, and G. Nágypál, "A reasoning framework for rule-based wsml," in *In Proceedings of 4th European Semantic Web Conference (ESWC)*, 2007.

 ISBN: 978-1-61208-152-6