Comparing Semantic Web Service Frameworks in a Context of Auction Services

Naoki Fukuta,[†] *and Takayuki Ito*[††]*,*

[†]Shizuoka University, Hamamatsu, Shizuoka, Japan
[††]Nagoya Institute of Technology, Nagoya, Aichi, Japan

Summary
In this paper, we compare Semantic Web Service (SWS) description framework WSMO and SWSF, and highlight their advantages for e-commerce services. The comparison mainly covers logical expression of services in the two frameworks. The contributions of this paper are 1) to evaluate the capabilities and the limits of those frameworks; 2)to clarify familiarity and mismatches of expected usage in SWS languages and combinatorial auction services; and 3)to investigate a novel usage of SWS technologies for e-commerce.

Key words:
semantic Web, auction mechanism, semantic Web services

Introduction

Automation mechanisms for building e-commerce systems are currently a hot topic. Automated Web service composition is a scenario for realizing such automation of building e-commerce systems by combining existing Web services. Semantic Web Services (SWS) has been proposed in [1] to realize automated composition of complex Web services by using semantic descriptions that are constructed using the Semantic Web technology. The advantage of using semantics in Web service composition is that we can compose services without any probabilistic approach and the composed services will run without semantic errors. To control these composition processes, rule languages are used to describe business rules and conditions for using services to realize valuable and economically valid service compositions.

To date, some important SWS frameworks and related technologies are proposed[2][3][4][5][6][7]. In this paper, We focus on SWSF[5] and WSMO[6][7]. Both frameworks have their own ontology and description languages for rules and logics that are needed for sufficient description of e-commerce services. SWSF means Semantic Web Service Framework, it includes the ontology SWSO(Semantic Web Service Ontology)[8] and the description language SWSL(Semantic Web Service Language)[9]. WSMO means Web Service Modeling Ontology. WSMO is an ontology but it includes the modeling language WSML(Web Service Modeling Language)[10] and the execution environment WSMX (Web Service eXecution Environment)[11]. WSMO has WSMF (Web Service Modeling Framework)[12] as its conceptual background. The rule language used in WSML has been submitted as an independent language WRL(Web Rule Language)[13]. Here, it is confusing that the structures and naming rules are different from SWSF and WSMO. In this paper, we use SWSF and WSMO, as the meanings of the whole frameworks[1].

Since the concepts, the ontologies, and the languages of these two frameworks are different, it is meaningful to compare the two frameworks and to understand the differences and advantages of them.

Although comparisons between SWSL and WSML are provided briefly within the W3C submissions of SWSF[5] and WSMO[6], the focus is on distinguishing the differences of purposes and approaches between the two; the W3C submissions do not provide an actual comparison in a certain business application context in the submission documents. However, it is valuable to understand which language is better suited for what types of business contexts. To the best of our knowledge, no comparison has been performed for expressiveness of SWSF and WSMO in the same e-commerce context.

In this paper, we compare the two frameworks, SWSF and WSMO in a context of using an auction mechanism. Here we focus on a comparison among logical expressions in two important conditions used for the winner determination problem in combinatorial auctions.

The rest of this paper is organized as follows. In Section 2, we show the relationships between SWS and auction mechanisms. Section 3 explains how to describe an auction and bids in the ontological expression used in SWS. In Section 4, we compare two different SWS languages with respect to logical expressiveness in the context of using services for determining winners for combinatorial auctions. We also briefly discuss the

[1] Recently, the WSMO team began to use *W<Triple>* as the name of whole framework including WSMO, WSML, WSMX, and additionally Triple Space Computing. We did not use this term in this paper since it did not appear in the W3C submissions.

Manuscript received April 5, 2006.
Manuscript revised April 25 , 2006.

comparison results presented in Section 4. Finally, in Section 5 we present our conclusions and outline future work.

2. Relationships between SWS and auction mechanisms

2.1 SWS and auction

Semantic Web Services(SWS) technology has a potential to enhance auction mechanisms for making contracts among agents by using semantics. Using auctions is a good candidate for making (semi-)optimal contracts in a simple and open way. However, in most current auction mechanisms, a bid is represented as an assignment of price for one item or combination of items. It is possible to assign bids to semantically represented items such as `a flight from Tokyo to Vienna'. Also, by using semantic information, some additional application might be possible, such as using trust information about bidders for assigning items in an auction.

Furthermore, there exist potential needs for using auction services in SWS Systems. There are plenty of online auction services on the Web and these auction services can be used in composing new services within an SWS system. For example, it is possible to implement intelligent multiple-bidding system like BiddingBot system[14] as an SWS system: it will obtain the desired items from multiple auction sites in lower price.

Here, an important issue will arise to combine SWS technology and auction mechanism: how do we express items, bids, sellers, buyers, auction processes, conditions to determine winners in auctions, etc. in a Semantic Web framework.

2.2 Implicit assumptions on the current major SWS frameworks

Through our survey, we understand that most SWS frameworks have a certain implicit assumption regarding processing tasks (in other words, queries). That is ``All tasks processed at the SWS execution engine are independent of each other, though these tasks should be processed separately."

This assumption is suitable for client-side applications such as finding and obtaining a desired item from auction sites. In a client side application, the goal that represents the task (in the above scenario, the item we want to obtain is the goal) is given by the user. Here, other users' goals (such as other people's color preference for the item) should not be reflected in the goal. Therefore, the assumption will work well in this situation. It is also possible to apply this assumption to server-side applications when the requests (goals) are not closely related each other and can be solved independently.

This assumption is also good for keeping an SWS framework and its implementation simple and easy to monitor and handle. This assumption is easily adopted for the event-driven architecture that is often used for implementing SWS execution environments.

This assumption does not however cover the situation where we need to solve multiple requests (goals) from multiple users when the requests are strongly related to each other.

This assumption in particular is incompatible with the initial motivation for multi-agent problem (task) solving, which is how to solve closely related problems (constraints) optimally. This mismatch will cause a problem when we use auction services on SWS frameworks. A contradiction is that we need all queries (in this case, a query is a bid assigned for an item by a bidder, or an item proposed for sale by a seller) to solve the problem (in this case, to determine who are the winners of the auction). However, each query is treated as an independent one and there is no way to know about other queries. In consequence, the problem will not be solved (and the winners of the auction will not be determined) while keeping this assumption.

The most important problem is that requests (goals) for SWS systems are so often represented as temporal messages just like an invocation parameter of a Web service. These temporal messages are not persistent therefore we need a storage mechanism for these messages in or outside of the SWS system.

2.3 Treating auction services as winner determination services

Here, we propose a solution for the problem pointed out in Section 2.2, which is simply treating auction services as those for determining auction winners.

From the viewpoint of the bidding process, there are two major process types in auction mechanisms. One is ``interactive" auctions, which are constructed in multiple rounds to interact to bidders. Ascending (English) auction is an example of interactive auctions. In interactive auctions, bidders can update their bids after the temporal bidding prices for items are opened. The auction will end when the conditions for closing are met, such as there being no updates of bids. In contrast to interactive auctions, ``one-shot" auctions are used in Vickley Auctions, Combinatorial Auctions, etc. In one-shot auctions, there is no chance for bidders to get information about other bids until the winners are determined. There is only one chance to place a bid, though in some cases the bid prices will not be the price to pay (Vickley Auction), or it may be possible to place multiple patterns of bids at the same time (Combinatorial Auction). In this paper, for simplicity we only consider one-shot auctions. Note that, in many cases we can extend one-shot auctions to the equivalent interactive auctions by employing proxy bidding or other appropriate mechanisms. In this paper, Combinatorial Auction[15] is used as an example of one-shot auctions.

Also, we assume that all bids for an item are gathered and stored in a place that can be accessed from the SWS system. This is possible when we use triple-space computing[16], which provides a persistent message storing mechanism for SWS.

To date, the idea of triple-space computing has only been proposed for WSMO-based frameworks. In this paper, however, we assume that this mechanism is available for all SWS frameworks. Extending the process of collecting bids in the auction is also possible by extending triple-space computing with certain user-interaction capabilities.

Here, SWS mechanisms are used to realize services that employ external auction winter-determination services. Selection of appropriate auction mechanisms is done in the SWS system by using descriptions of features of auction mechanisms. Here, a remaining problem is how to describe features of various auction mechanisms in those SWS languages.

3. Ontological expression of a combinatorial auction

3.1 List representation vs. triple-based representation

Before beginning an exploration of detailed auction descriptions, we need to understand the existence of a gap between the logical definition of auctions and Description Logic(DL)-based representation used in the Semantic Web world.

In the realm of Web Services world (in other words, in the world of ordinal programming), multiple variable-length data are typically treated as an ordered list. Here, input bid sets and winners' bid sets are often represented as ordered lists. Here, an important point is that such lists also implicitly contain meta-data such as shared attributes or relations of containing data. For instance, a list for input has implicit meaning that the containing data is input bid set, and a list for output has implicit meaning that the containing data will be a bid set of winners'.

In SWS world, all information is treated as knowledge (assertion) and that is represented as triples, the standard representation form on the Semantic Web. In this world, we should not use list representation used in the Web Services world. Instead, we just represent such implicit attributes or relations as triples. So we need to have a form and conversion to represent them in triple format. For example, a bid should have a relation to a certain bid set, and may have a relation to a bid set of winners' on a certain auction.

3.2 Triple-based representation of bid sets

In this section, we provide an example of triple-based representation of bids, bid sets and auctions. All those data are represented as resources, attributes and relations. Here we do not distinguish between attributes and relations, we just use properties that are normally used in Semantic Web languages such as OWL and RDF.

There is a bid set X and a bid Y. The bid set X has a property includedIn that represents a bid included in this bid set. Here, we denote that the bid set X has a property includedIn and the value is a certain bid, for example, a bid Y. In pseudo-triple format, the example is denoted as hasValue(id_of_X, includedIn, id_of_Y) .

When a bid set X is a winners' set in a certain auction Z, we denote it in the same way using a property hasBidSetOfWinners of the auction Z, by using a triple hasValue(id_of_Z, hasBidSetOfWinners, id_of_X). Here, we can infer that a bid Y is a winner's bid and we can obtain the item(s) won by the bid Y.

The question may arise that ``Here we have only one bid in a bid set X. Is this really valid?''. In response, we introduce the concept of **cardinality**, which is often used in the Semantic Web world: A resource possesses two or more properties that have same name, but they should be lower than, or greater than, the provided cardinality constraints in the ontology. Here, we give a cardinality constraint that a bid set has one or more properties of includedIn. We will show how such cardinality constraints are represented in an ontology definition later. Now we can represent that the bid set X has other bids T, U, and V, by using triples such as hasValue(id_of_T, includedIn, id_of_X), hasValue(id_of_U, includedIn id_of_X), and hasValue(id_of_V, includedIn, id_of_X). These properties are stored and used in the knowledge base of a SWS system.

3.3 Ontology for representing bid sets

In Figure 1, we provide a concrete example definition of an ontology to represent bid sets, bids, and auctions. Here, for better readability, we use WSML as the ontology definition language rather than OWL. The definition below can be easily converted to another major ontology language such as OWL.

Here, we give example instances of auctions and bids in Figure2.

Note that we provided only essential parts of our ontology here. For example, in real usage we need a more detailed ontology and instances for items and owners, but those parts are omitted in this paper.

```
concept bid
    // has one or more items
    hasItem impliesType (1 *) item
    // just one bid price for the items
    hasBidPrice ofType (1) _integer
    hasOwner impliesType (1) owner
  concept bidSet
     // has one or more bids
    hasBid impliesType (1 *) bid
  concept combinatorialAuction
    hasInitialBidSet impliesType (1) bidSet
    hasBidSetOfWinners
      impliesType (0 1) bidSet
    hasItem impliesType (1 *) item
  concept item
    hasName ofType _string
    hasOwner impliesType (1) owner
  concept owner
    hasName ofType _string
```

Fig. 1 Combinatorial Auction Ontology written in WSML

```
instance auction01
  memberOf combinatorialAuction
  hasInitialBidSet hasValue bidSet01
  hasItem item01
  hasItem item02
  hasItem item03
instance bidSet01 memberOf bidSet
  hasBid hasValue bid01
  hasBid hasValue bid02
  hasBid hasValue bid03
instance bid01 memberOf bid
  hasItem item01
  hasItem item02
  hasBidPrice 310
  hasOwner owner01
instance bid02 memberOf bid
  hasItem item03
  hasBidPrice 100
  hasOwner owner01
instance bid03 memberOf bid
  hasItem item01
  hasItem item03
  hasBidPrice 3000
  hasOwner owner02
```

Fig. 2 Example of instances defined by the combinatorial auction ontology

4 Example descriptions of combinatorial auction service

4.1 Basic model of an auction-winner determination service

First of all, we present a (process) model for an auction-winner determination (AWD) service of combinatorial auctions. An AWD service has a bid set as input and will produce a bid set as an output. The output bid set should follow the two conditions: the partitioning condition and (optionally) the covering condition. The output bid set should also be a (semi-)optimal solution to maximize the total utility of sellers. The AWD service's process is just to have an input bid set and obtain the resulting bid set as output. An AWD service may include further processes to register items for the auction, or to find items that are related to a buyer, etc. Here, we omit those additional processes from the process of AWD service to keep the descriptions of the process (we will show them later) simple.

The two conditions are described in logical formulae as follows.

(1) Partitioning condition

Let *M* be the set of items to be auctioned. Then any bidder, *i*, could place any bid b_i *(S)* for any combination $S \subseteq M$.

Let *X* be a valid outcome, an outcome where each item is allocated to only one bidder:

$$X = \{S \subseteq M \mid \exists b\ (S), i \in bidders, and$$
$$S \cap S' = \phi \quad for\ every\ S, S' \in X\}$$

(2) Covering condition

The covering condition means that all possible items should be sold:

$$\bigcup_{x \in X} x = \bigcup_{m \in M'} m$$
$$such\ that\ M' = \{S \subseteq M \mid \exists b_i(S), i \in bidders\}$$

In the next two sections, we will explain how these logical expressions can be represented in the two major service description languages used in WSMO and SWSO.

4.2 Example in WSMO

WSMO (Web Service Modeling Ontology) is a set comprising an ontology and a description language for composing Web Services. WSMO is based on WSMF[12]. It has been submitted to W3C to be discussed in relation to the next-generation semantic Web services standard.

Figure 3 shows an example description of the AWD service in WSMO. In WSMO, Web services are described in four parts: the ontology to be imported, mediators to be used, capability of the service, and the interface. In the context of the AWD service, only one simple process is considered and no mediator is used here. The interface part of WSMO is mainly used for multiple complex processes but not for representing the input and output parameters of each service process. Thus, we focus on presenting the capability part, omitting the interface part.

Notice that in WSMO, a service is NOT modeled as a function that has certain input and output parameters, rather, the input parameters are provided through the current state of the knowledge base and the output will be reflected in updates of the knowledge base. Therefore, the service's capability is modeled as the conditions in the knowledge base which should be satisfied before, through, or after the service invocation.

In the example, the precondition is just checking for the presence of a target bid set for determining winners, and the assumption and effects are always true since we do not consider a real buying process that includes payment by credit card, etc.

The most important part here is the postcondition part of the capability. In the postcondition part, we provide two conditions to be satisfied after invocation of the AWD service. Here, we employ logical expressions in WSML to describe those two conditions. In WSML, F-logic level logical expressions can be used. Note that, roughly speaking, the F-logic is a combination of first-order logic formulae and frame-based descriptions for objects that have slots and slot values. Here, variables are noted as identifiers that start with the '?' character. The frames are denoted using '[' and ']'. For example, the notation ?somebid[hasItem ?item] represents that the instance indicated the variable ?somebid has a slot hasItem and its value is bound to the variable ?item. Here we can use forall, exists, and implies operators to describe the conditions since the description language allows first-order description without any limitation. The notion of naf means '**Negation As Failure**', that treats negation as the failure to find the satisfied conditions.

```
webService WinnerDeterminationService~
ForCombinatorialAuctionsWebService
 importsOntology
   _"http://example.org/Ontology/Combinatorial~
AuctionsOntology"
 capability WinnerDeterminationServiceFor~
CombinatorialAuctionsCapability
   sharedVariables{?resultSet,?inputSet}
   precondition
     definedBy ...
   assumption
     definedBy ...
   postcondition
     definedBy
       // partitioning condition
       forall {?bid, ?anotherBid}
               (?resultSet[
                   hasBid hasValue ?bid,
                   hasBid hasValue ?anotherBid ]
                 and
                 naf ( ?bid equivalent ?anotherBid )
                 implies
                 forall {?x, ?y}
                 (?bid[ hasItem hasValue ?x ]
                  and ?anotherBid[ hasItem hasValue ?y ]
                  implies
                  neg (?x equivalent ?y)))
       and
       // covering condition
       forall {?item}
               (exists {?somebid} ?inputSet[
                   hasBid hasValue ?somebid ]
                and ?somebid[ hasItem ?item ]
               implies
               (exists {?bid}
                (?resultSet[ hasBid hasValue ?bid ]
                 and ?bid[ hasItem hasValue ?item ])).
   effects
     definedBy ...
```

Note: the line which ends with char '~' means that the line continues to next line without any spaces. This is not the ordinal syntax of WSML, only used for this figure.

Fig. 3 Example Description of AWD Conditions inWSML

4.3 Example in SWSO

In contrast to WSMO, SWSO uses the concept of input and output of services explicitly. In SWSO, conditions are separately described (defined) and attached to the input or output of a service. Below is an example of the AWD service in SWSO. Here, we only show the process part; other parts such as profile and grounding are omitted. The AWD service is modeled as a simple service that has a certain input and output. Note, however, that we treat the input and output as only IDs that indicates the input and output. Actual information about the input and output are represented as resources that have certain relations to the IDs. This is because it is very difficult to clearly distinguish between the input, the output and existing information. For example, in ordinal Web-based auction sites, the name of an item already exists before the bidding begins, but the resulting output may contain this name as duplicate information. This is useful for avoiding mistakes such as identifying two different items as the same one. However, when the service is ready for semantic Web, these identification problems will not occur. Therefore, we only need certain IDs to determine the input and output data when all data and services become semantic Web ready. In WSMO, this idea is deeply embedded into the modeling process, but in SWSO, it can be used but is not be a necessary requirement for modeling services.

The example description shown in Figure 4 is in a human-readable format. The actual description of it will be like a triple-based representation in the RDF format. Since SWSO only allows condition descriptions for outputs as conditional outputs, here, we used a small trick to represent the output's postcondition. The predicate get_winner_allocation/2 invokes the actual winner-determination service and obtains the resulting winners, but it will only be the final output when the two conditions (partitioning_condition/1 and covering_condition/2) are satisfied. The actual definitions of these two conditions are described following the service process definition independently. The definitions of conditions are described in Prolog-like format, but, will be encoded in RuleML-OWL or another appropriate format when in the actual use. The notions of variables are just same as in WSMO, where IDs starting with the '?' character denotes variables. The notion of naf means '**Negation as Failure**', the meaning of which is equivalent to the same notion in WSMO. Note that we use a predicate equivalent/2 to determine whether two IDs are equal instead of using the same variable name for them, since two different IDs may point to the same thing(resource, in the term of OWL/RDF) in the OWL ontology description format. Here we do not use full-spec F-Logic description but employ the Horn logic layer instead. Consequently, the condition descriptions are very simple and easy to understand for ordinal Prolog programmers. (This expression is called SWSL-Rules. Note that it is possible to use a full-spec F-Logic format here. This is called SWSL-FOL. It is not possible to use these two different description languages in a same description: a special bridge description is additionally required with two different descriptions separately. This limitation does not exist in WSMO. In WSMO, it is possible to use different layers of WSML descriptions in a document seamlessly.)

```
determine_combinatorial_auction_winners {
   Atomic
   input input_bidset_id
   output ( get_winner_allocation(
            input_bidset_id,result_bidset_id),
  partitioning_condition(result_bidset_id)
  and covering_condition(
         input_bidset_id,result_bidset_id) ),
  winner_allocation(
         input_bidset_id,result_bidset_id)
}

// below are definitions of conditions used above in
// `pretty print' format

 partitioning_condition(?ResultBidSet) :-
        naf partitioning_condition_violation(
             ?ResultBidSet)

partitioning_condition_violation(?ResultBidSet) :-
        hasBid(?ResultBidSet,?Bid) and
        hasBid(?ResultBidSet,?AnotherBid) and
        naf equivalent(?Bid,?AnotherBid) and
        hasItem(?Bid,?X) and
        hasItem(?AnotherBid,?Y) and
        equivalent(?X,?Y)

covering_condition(?InputBidSet,?ResultBidSet) :-
        naf
      convering_condition_violation(?InputBidSet,
          ?ResultBidSet)
convering_condition_violation(?InputBidSet,
                                  ?ResultBidSet) :-
        hasBid(?InputBidSet,?Bid) and
        hasItem(?Bid,?Item) and
        hasBid(?ResultBidSet,?ResultBid) and
        naf hasItem(?ResultBid,?Item)
```

Fig. 4 Example Description of AWD Conditions in SWSL

4.4 Comparison of descriptions in WSMO and SWSF

Table 1 shows the comparison between WSMO and SWSF in four aspects: expressiveness of rule and logic, unified logical descriptions, expressiveness of formal processes, and controllability of process execution.

WSMO and SWSF are not very different from the perspective of how they describe services: They both have ontological expressions and allow logical expressions for behaviors of services. Both languages are based on a layered approach, that comprises several layers, each of which has a different level of expressiveness.

WSML, the language for WSMO has 5 layers, named WSML-CORE, WSML-DL, WSML-Flight, WSML-Rule, and WSML-Full. In WSML, a stacking approach is used so that WSML descriptions are seamlessly extended to higher-layer expressions. In contrast, SWSL uses a branch approach with stacking. In SWSL, there are two independent description language lines, SWSL-FOL and SWSL-Rule(In [5], SWSL-FOL is treated as a subset of SWSL-FOL, but these two languages have different interpretations of the same expression. Therefore, they cannot be used in a same document or fragment of it. Thus we argue it is better to treat the two languages as two independent languages.) Although the two languages share many portions of the syntax, the underlying semantics are slightly different. In SWSL, compatibility of descriptions of classical rule languages is very highly prioritized. This choice causes a semantic incompatibility between SWSL-FOL and SWSL-Rule. In SWSL, the separate descriptions of two languages and the use of a bridge description between them is recommended. Although the SWSL's approach has an advantage in describing rules for current rule-description specialists, it will cause some frustration and confusion for newcomers.

Process descriptions in WSMO and SWSO are quite different. In WSMO, a process is described as a multiple-state-machine in which state transitions are controlled by logical (and procedural) expressions in WSML, so it more closely resembles programming rather than ontological definitions of a process. Therefore, WSMO features good controllability in expressing actual executions of service processes. The control flow of a process in SWSO, on the other hand, is more ontological, and it follows a concurrent computation theory. In SWSO, a process is formulated by pi-calculation. Since a process in SWSO may contain branches, splits and joins of two or more concurrent sub processes, etc, SWSO has higher-level expressiveness for describing processes, but pays less attention to the actual executions of services.

Table 1: A comparison chart of WSMO and SWSF

	WSMO	***SWSF***
Expressiveness of rule and logic	**	***
Unified logical descriptions	***	*
Expressiveness of formal processes	*	***
Controllability of process execution	***	*

5 Conclusions

In this paper we compared two next-generation SWS framework -- SWSF and WSMO -- from the perspective of describing logical expressions for auction winner determination services. Two examples demonstrated that both languages are sufficiently expressive to represent two conditions for winner determination in a combinatorial auction. Through our comparison, we found that one advantage of SWSL is its compatibility with legacy rule description formats, making rule description easy for users who are specialists in existing rule languages such as Prolog. The advantages of WSMO are that it effectively controls process execution and seamlessly describes of different syntax layers for F-Logic and rules.

A possible future work is how do we describe reputation information in WSMO and SWSF. In current e-auction systems, using reputation information about bidders and sellers is essential. Currently, reputation information is represented as an integer number, and the users themselves take into account such reputation in their information in their bidding. Once agent-based proxy auctioning becomes predominant, it is likely that information that reflects reputation will be considered by bidder and seller agents. By using Semantic Web technology, agents will be able to infer much more about the reputation of buyers and sellers for their bidding processes.

It is also possible to consider reputation information in the AWD process itself. For example, if one bidder has won an item but will not pay for it, it is possible to re-assign the item to another bidder. This mechanism is already implemented in some existing e-auction systems. Rejecting such irrelevant bids to prevent item assignment to inappropriate winners can be included in next-generation AWD mechanisms by using Semantic Web technology. In such cases, it will also be a challenge to represent and describe such reputation information in the world of the Semantic Web.

References

[1] McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Intelligent Systems **16** (2001) 46—53

[2] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K., OWL-S: Semantic Markup for Web Services. W3C Member Submission 22 November 2004 (2004) http://www.w3.org/Submission/OWL-S/

[3] Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., Verma, K., Web Service Semantics - WSDL-S, W3C Member Submission 7 November 2005 (2005) http://www.w3.org/Submission/WSDL-S/

[4] Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar S., and Miller, J., METEOR-S WSDI: A Scalable

Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers.

[5] Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., Su, J., Tabet, S.: Semantic web services framework (swsf) overview. W3C Member Submission 9 September 2005 (2005) http://www.w3.org/Submission/SWSF/.

[6] de Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., Konig-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., Stollberg, M.: Web service modeling ontology (wsmo). W3C Member Submission 3 June 2005 (2005) http://www.w3.org/Submission/WSMO/.

[7] Roman, D., Keller, U., Lausen, H., de~Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web service modeling ontology. Applied Ontology **1** (2005) 77--106

[8] Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., Su, J., Tabet, S.: Semantic web services framework (swso). W3C Member Submission 9 September 2005 (2005) http://www.w3.org/Submission/SWSF-SWSO/.

[9] Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., Su, J., Tabet, S.: Semantic web services language (swsl). W3C Member Submission 9 September 2005 (2005) http://www.w3.org/Submission/SWSF-SWSL/.

[10] de Bruijn, J., Fensel, D., Keller, U., Kifer, M., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L.: Web service modeling language (wsml). W3C Member Submission 3 June 2005 (2005) http://www.w3.org/Submission/WSML/.

[11] Bussler, C., Cimpian, E., Fensel, D., Gomez, J.M., Haller, A., Haselwanter, T., Kerrigan, M., Mocan, A., Moran, M., Oren, E., Sapkota, B., Toma, I., Viskova, J., Vitvar, T., Zaremba, M., Zaremba, M.: Web service execution environment (wsmx).W3C Member Submission 3 June 2005 (2005) http://www.w3.org/Submission/WSMX/.

[12] Fensel, D., Bussler, C.: The web service modeling framework wsmf. Electronic Commerce Research and Applications **1** (2002) 113--137

[13] Angele, J., Boley, H., de~Bruijn, J., Fensel, D., Hitzler, P., Kifer, M., Krummenacher, R., Lausen, H., Polleres, A., Studer, R.: Web rule language (wrl). W3C Member Submission 9 September 2005 (2005) http://www.w3.org/Submission/WRL/.

[14] Ito, T., Fukuta, N., Shintani, T., Sycara, K.: Biddingbot: A multiagent support system for cooperative bidding in multiple auctions. In: Proc. of the Fourth International Conference on Multi Agent Systems (ICMAS'2000). (2000) 399--400

[15] Cramton, P., Shoham, Y., Steinberg, R.: Cmobinatorial Auctions. The MIT Press (2006)

[16] Fensel, D.: Triple-space computing: Semantic web services based on persistent publication of information. In: Proc. of Semantic Web Services: Preparing to Meet the World of Business Applications -- a Workshop at the 3rd International Semantic Web Conference(ISWC2004)--. (2004)

Naoki Fukuta received B.E. and M.E. from Nagoya Institute of Technology in 1997 and 1999 respectively. He received Doctor of Engineering from Nagoya Institute of Technology in 2002. Since Apr. 2002, He has been working as a research associate at Shizuoka University. His main research interests includes Mobile Agents, SemanticWeb, Knowledge-based Software Engineering, Logic Programming, and WWW-based Intelligent Systems. He is a member of ACM (Association for Computing Machinery), IEEE-CS(IEEE Computer Society), JSAI (Japanese Society for Artificial Intelligence), IPSJ (Information Processing Society of Japan), IEICE (Institute of Electronics, Information, and Communication Engineers), JSSST (Japan Society of Software Science and Technology), and ISSJ(Information Systems Society of Japan).

Takayuki Ito received the B.E., M.E, and Dr. of Engineering from the Nagoya Institute of Technology in 1995, 1997, and 2000, respectively. From 1999 to 2001, he was a research fellow of the Japan Society for the Promotion of Science (JSPS). From 2000 to 2001, he was a visiting researcher at USC/ISI (University of Southern California / Information Sciences Institute). From 2001 to 2003, He was an associate professor of the Center for Knowledge Science in Japan Advanced Institute of Science and Technology (JAIST). He joined the Division of Computer Science and Engineering, the Graduate School of Engineering, Nagoya Institute of Technology in April 2003. He is a Founder, a Senior Vice President, Chief Operating Officer of Wisdom Web Co., Ltd. from July 13, 2004.From 2005 to 2006, he is a visiting scholar at Faculty of Art of Science, Division of Engineering and Applied Science, Harvard University. From Sep. 2005 to Feb. 2006, he is a visiting researcher at Center for Coordination Science, Sloan School of Management, Massachusetts Institute of Technology. He also joined the Master Course of Techno-Business Administration (MTBA), Nagoya Institute of Technology from April 2006. His main research interests include Multi-Agent Systems, Computational Mechanism Design, Game Theory, Auction Theory, Intelligent Agents, Distributed Artificial Intelligence, Computational Biology, Bioinformatics, Group Decision Support Systems, Agent-mediated Electronic Commerce, Information Economics, and Reasoning under Uncertainty. He is a member of AAAI(American Association for Artificial Intelligence), ACM(Association for Computing Machinery), JSAI(Japanese Society for Artificial Intelligence), IPSJ(Information Processing Society of Japan), IEICE(Institute of Electronics, Information, and Communication Engineers), JSSST(Japan Society for Software Science and Technology), and SICE(Society of Instrument and Control Engineers).