Incorporating Situation Awareness in Service Specifications

Stephen S. Yau and Junwei Liu
Arizona State University
Tempe, AZ 85287-8809, USA
{yau, junwei.liu}@asu.edu

Abstract

Service-Oriented Architecture has the major advantage of enabling rapid composition of distributed applications from various service*s, and has become increasingly popular for many large-scale service-based systems in various application areas, including scientific collaboration, e-business, health care, military, and homeland security. Situation awareness (SAW) is the capability of the entities in a service-based system to be aware of the situation changes and automatically adapt themselves to such changes to satisfy user requirements, including security and privacy. The continuing evolutions of the entities and environment makes SAW one of the most desired features to support dynamic adaptive computing in service-based systems. In this paper, the relationship between contexts/situations and services in situation-aware service-based systems is identified and an extension of OWL-S with situation ontology, called* SAW-OWL-S, *incorporates SAW in service specifications is presented. An approach to generating service specifications for situation-aware service-based systems using SAW-OWL-S and the system diagram of situation-aware service-based systems using SAW-OWL-S are presented.*

Keywords: Service-oriented architecture, situation awareness, service specification, web ontology language for Web services, service-based systems.

1. Introduction

Service-based systems are based on *Service-Oriented Architecture* (SOA) [1], which has become increasingly popular for many large-scale service-based systems in various application areas, such as scientific collaboration, e-business, health care, military, and homeland security. Service is considered as a software/hardware entity with well-defined interfaces to provide certain capability over heterogeneous platforms. The major advantage of SOA is its capability of rapid composition of distributed applications from various services, regardless of the programming languages and platforms used in developing and running different components of the applications.

In a service-based system, the entities and environment all evolve dynamically, and the entities in the system need to adapt themselves to such evolution to achieve user objectives. *Situation awareness* (SAW) is an important feature to support such dynamic adaptive service-based computing. A *situation* is a set of contexts in the application over a period of time that affects future system behavior. A *context* is any instantaneous, detectable, and relevant property of the environment, the system, or users, such as location, available bandwidth and a user's schedule. SAW is the capability of the entities in service-based systems to be aware of the situation changes and automatically adapt themselves to such changes [2, 3]. Situation-aware service-based systems are more dynamic and flexible to satisfy user requirements, including security and privacy, than traditional service-based systems since service discovery, service access and service execution can adapt to situation changes.

To support situation-aware service-based systems, both service specification and related context and situation information need to be clearly specified and shared among various entities of the systems. To achieve this goal, the relationship between contexts/situations and services needs to be first identified and the incorporation of SAW in service specifications needs to be addressed. In this paper, we will identify the relationship between contexts /situations and services in situation-aware service-based systems, and an extension of OWL-S with situation ontology, called SAW-OWL-S, that incorporates SAW in service specifications will be presented. An approach to generating service specifications for situation-aware service-based systems using SAW-OWL-S will then be presented, and an example is given to illustrate this approach. The system diagram of situation-aware service-based systems using SAW-OWL-S will also be presented.

2. Relationship between Contexts /Situations and Services

Most of traditional service specification approaches, such as Web Ontology Language for Web Services (OWL-S, formally DAML-S) [4] and Web Services Description Language (WSDL) [5], describe the service information in three categories: what is the service? How to use it? How does it work? To support SAW, services in service-based systems should be able to adapt their behavior according to different situations, which requires specifying contextual data of services, the preconditions of services, the effects of services and the triggering rules of services. Let us consider the following scenarios to show the importance of the relationship between context/situation and service:

Scenario1. A user wants to print a document on a nearby printer with least pending tasks.

Scenario2. A printer service requires that only the user nearby can access it.

Scenario3. User A prints some document on a printer for user B, after the printing job is done, user B should know the document is ready for pickup.

Scenario4. When the presentation is finished, the light in the room should be automatically turned on.

Scenario1 is a very common service discovery problem. To evaluate whether a printer service is nearby, the location of the printer must be described in the service specification and the function of "*nearby*" needs to be clearly defined. Furthermore, since there are various entities in a service-based system, it is necessary that the description of the location and "*nearby*" must have formal semantics which can be understood by different parties in the system. With such formal semantics, although two different parties in the system may have different definitions of "*nearby*", they can still understand each other.

Besides the static location contextual data, the amount of pending tasks of the printer service dynamically changes. One approach to addressing this is that the printer service provides a process which can return the current pending tasks upon request. However, this increases the complexity of the printer services. A more flexible approach is to describe the current pending tasks as a context of the printer service and let the context management component take care of the update of the contextual data. Using this approach, when the user's request requires certain contextual data of the service, the system only needs to check whether a contextual data satisfying the user's request is associated with the service. New contextual data, such as current memory usage of the printer and the bandwidth of the connecting link of the printer, can be introduced during runtime.

In Scenario2, the service has a pre-condition defined on contextual data of the user. For a traditional service specification approach, the pre-condition of the process is defined on the input and internal parameters of the process. We denote the pre-conditions which are defined on external data of the service as *external pre-condition*. Although the user can pass the location information as input into the printer service, this increases the printer service's complexity and does not support dynamic adaptation. A better solution is to specify a situation of "nearby of printer's location" as the pre-conditions of the service. This will not increase the service's complexity and if the definition of the pre-conditions is changed, only the service specification needs to be changed.

The execution of services will also affect the system, such as in Scenario3. We denote such a kind of effect as service's *external post-conditions*. Similar to service's external pre-conditions, we can specify such external post-conditions of services using situations.

Finally, a service may also be triggered by certain situations like in Scenario4. The service specification should include the rule like "the service will be triggered under what situations". According to such specification, situation-aware service-based systems can perform the triggering action when the trigger situation is satisfied.

Based on the analysis of these scenarios, the following are the four main relations between contexts/situations and services, which need to be modeled in a service specification for situation-aware service-based systems:

R1. Service's contextual data: For each service, it may have associated contextual data, which can be used to determine whether certain situations are satisfied.

R2. Situation pre-condition: The service may require certain external pre-condition to be satisfied to execute the process.

R3. Situation post-condition: The execution of the service may result in certain external post-conditions.

R4. Situation-service-triggering: The situation can also be defined as the trigger of certain services.

3. Current State of the Art

Several specification languages have been developed for specifying Web services. Among them, WSDL and OWL-S are most popular. WSDL is an

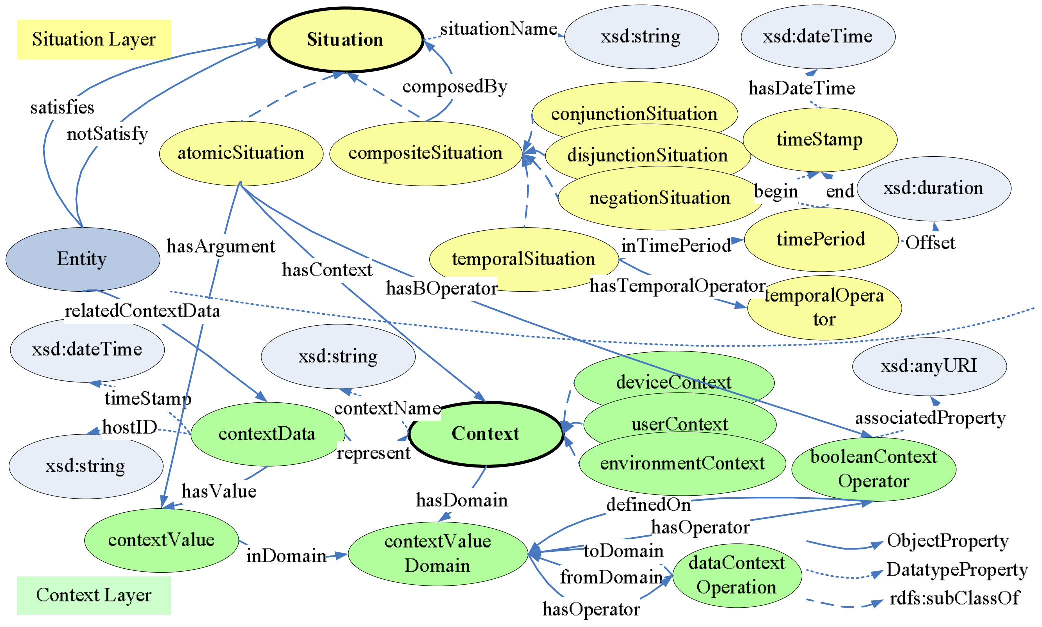

Figure 1. The OWL-based situation ontology

XML-formatted language used to describe the capabilities of a Web service as collections of communication endpoints capable of exchanging messages. WSDL provides a basic and simple abstraction of Web services. It is a contract or complete description that describes the components being exposed, and provides the names, data types (using XML Schema Definition), methods, and parameters required to call them. The overall structure of OWL-S includes three main parts: the service profile for advertising and discovering services; the process model, which gives a detailed description of a service's operation, including the IOPE (Input, Output, Precondition, and Effect) parameters of the process; and the grounding, which provides the details on how to interoperate with a service, via messages. OWL-S provides the primitives for service descriptions in semantic web. Both WSDL and OWL-S are widely used service specification standards. However, formalisms for expressing context and situation information are not supported in both WSDL and OWL-S.

To incorporate context awareness in service specification, the Aspect-Scale-Context (ASC) model and Context Ontology Language (CoOL) [6] can be plugged in the DAML-S model to represent the context information of services and enable context awareness and contextual interoperability during service discovery and execution. CWSDL (Context-Based Web Services Description Language) [7] aims at enhancing the actual service description language with context-aware features. The context-aware service discovery is also discussed in [7] by defining ContextFunction to collect the related contextual data of the service and electService to rank the services based on service discovery request and contextual data, including user preference. Another related approach [8] targets at service collaborating in business environments by defining a context for multiple entities to work with and share execution-specific data. Context attributes [9] are used to specify both static and dynamic contextual data of services. However, none of these approaches clearly define the relationship between contexts/situations and services in service-based systems and incorporate situation awareness in service specifications.

A conceptual model for context/situation and the relationship between contexts/situations and services for service-based systems and a situation specification example based on the conceptual model using F-logic are presented in [10, 11]. In [12], a hierarchical OWL-based situation ontology for situation modeling and reasoning is presented. Since OWL-S is a widely used service specification language, in this paper, we will show how to extend OWL-S with the OWL-based situation ontology [12] to incorporate SAW in service specifications.

For the sake of completeness, we will briefly describe the OWL-based situation ontology here. For detailed description of situation ontology, the reader is referred to [12].

As shown in Figure 1, the situation ontology models context and situation in a hierarchical approach such that the definitions for context and situation can be easily shared and reused. The situation ontology is extensible to user-defined domain specific situation knowledge. The situation ontology can be roughly divided to two layers: context layer and situation layer.

Context layer: The conceptual context definition, the realistic contextual data and the value of the contextual data are modeled using Context class, contextData class and contextValue class. Any Entity can specify associated contextual data using relatedContextData property. The context value belongs to contextValueDomain and context interpretation between different domains is represented as dataContextOperation. One special type of context interpretation that provides Boolean output is defined separated as booleanContextOperator. The booleanContextOperator is used to define situations.

Situation layer: The situation layer is build on top of the context layer to aggregate contextual data into situations. Different Situations form a hierarchy based on their derivation. The atomicSituation class represents all the basic situations whose value is directly derived from contextual data. The compositeSituation class represents more complicated situations: either the logical composition over other situations (conjunctionSituation, disjunctionSituation and negationSituation), or the temporalSituation whose value is derived from the value history of another situation. An Entity in the system may satisfies or notSatisfy a situation.

4 Incorporating SAW in OWL-S

As mentioned in Section 2, there are four main relations between contexts/situations and services, which need to be specified for services of situation-aware service-based systems. These situation-awareness related service specification requirements include service contextual data, situation pre-condition, situation post-condition and situation-service-triggering. These relations are also defined in the conceptual model in [10, 11].

In our approach, the situation ontology is incorporated in OWL-S to specify these situation-aware features for services. We denote the extended service specification ontology as SAW-OWL-S. Figure 2 shows the key classes and relations in SAW-OWL-S. The real-time timing aspects of service specifications are not considered in SAW-OWL-S. The integration includes following three aspects:

1. As an entity in situation-aware service-based systems, services may have associated contextual data. Based on such contextual data, whether certain situation is satisfied by a service or not can be determined. In SAW-OWL-S, the Service class

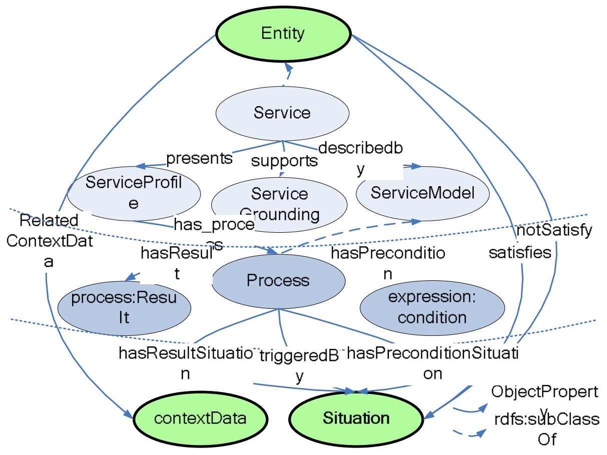

Figure 2. SAW-OWL-S (only related important classes and relations are shown)

   is defined as a subclass of the Entity class, so that the object property relatedContextData can be used to link related contextual data to the service and satisfies / notSatisfy can be used to specify whether the service satisfies a specified situation or not.
2. In OWL-S, the Process class models the internal model of a service, which already defines the pre-condition, post-condition, input, output and result of processes. However, the pre-condition and post-condition defined in OWL-S is based on the input or internal parameters of the process. In order to support external pre-conditions/post-conditions, two object properties called hasPreconditionSituation and hasResultSituation, whose ranges are Situation class, are added in the process class. The user can use these two properties to define the external pre-condition and post-condition relations between situations and processes.
3. In the process class, an object property, called triggeredBy, whose range is Situation class, is also added. The user can use this property to define the triggering relation between situations and processes.

5 Service Specification for Situation-aware Service-based Systems Using SAW-OWL-S

Our approach to service specification for situation-aware service-based systems includes three steps:

S1. Specify the traditional OWL-S service specification;

S2. Identify all the contexts and situations related to the service, which need to be specified. For each of the related contexts and situations, the user needs to either specify an OWL specification based on the situation ontology or finds an available one.

S3. Specify the related context/situation information of the service as follows:

S3.1. The contextual data of the service is specified using relatedContextData property of the service;

S3.2. The properties of the service hasPreconditionSituation, hasResultSituation and triggeredBy are then specified to the corresponding situations.

The context and situation specifications used in Step S2 can be shared and reused with multiple components in service-based systems. The system may have a repository of pre-defined context and situation specifications.

To illustrate this approach, consider the following scenario: There are five services involved in this scenario: presentationService (S1), lightTriggerService (S2), printerAService (S3), printerBService (S4) and videoPlayBackService (S5). Suppose that userA needs to give a presentation at a meeting. The presentation service can be accessed by userA only if userA is located in the meeting room and the light is dimmed for more than two seconds. When the presentation is finished, the full light should be automatically turned on. After the presentation, userA wants to print several reports with a printer on the same floor and only printerA satisfies this constraint. The meeting is recorded into video streams. After the meeting, userA can access the video stream of the meeting only if the device has broadband connections.

Following are the three steps in our approach,

1. The traditional OWL-S service specification for these services is first generated;
2. The four contexts involved in the five services are locationContext, lightContext, presentationStatus-Context and networkBandwidthContext. Based on these contexts, various situations are defined as shown in Table 1.
3. Specify the relations between the context/situation information and services as shown in Table 2.

6. SAW-OWL-S Situation-aware Service-based Systems

With the SAW-OWL-S, SAW can be incorporated into service specifications of traditional service-based systems. In this section, we will present an overall system diagram of SAW-OWL-S situation-aware service-based systems, logic inferences on SAW-OWL-S specifications and the interactions in SAW-OWL-S situation-aware service-based systems.

Table 1. Specification of situations

Situation	Definition
(Sit1) *LightOff* (atomicSituation)	hasContext: *lightContext* hasBOperator: *sameAsOp* hasArgument: *falseValue*
(Sit2) *LightOff3S* (temporalSituation)	composedBy: *LightOff* hasTemporalOperator: *alwaysTrue* inTimePeriod: *Past3S*
(Sit3) *LightOn* (atomicSituation)	hasContext: *lightContext* hasBOperator: *sameAsOp* hasArgument: *trueValue*
(Sit4) *InConferenceRoom* (atomicSituation)	hasContext: *locationContext* hasBOperator: *sameAsOp* hasArgument: *crLocation*
(Sit5) *ReadyForPresentation* (conjunctionSituation)	composedBy: *LightOff3S* composedBy: *InConferenceRoom*
(Sit6) *PresentationFinished* (atomicSituation)	hasContext: *presentationStatusContext* hasBOperator: *sameAsOp* hasArgument: *trueValue*
(Sit7) *OnSameFloorAsUserA* (atomicSituation)	hasContext: *locationContext* hasBOperator: *sameFloorOp* hasArgument: *userALocation*
(Sit8) *HasBroadband-Connection* (atomicSituation)	hasContext: *networkBandwidthContext* hasBOperator: *greatThanOrEqualToOp* hasArgument: *256KB*

Table 2. Specification for the Relationship between Contexts/Situations and Services

	S1	S2	S3, S4	S5
hasPrecondition-Situation	Sit5			Sit8
hasResult-Situation	Sit6	Sit3		
triggeredBy		Sit6		
Related-ContextData			locationA, locationB	

6.1 SAW-OWL-S Situation-aware Service-based System Diagram

SAW-OWL-S supports the specification of related context/situation information of services. In Figure 3, the system diagram of a situation-aware service-based system based on SAW-OWL-S is illustrated. At the center of Figure 3, service provider, service requestor and service directory in the traditional SOA remain the core of a situation-aware service-based system.

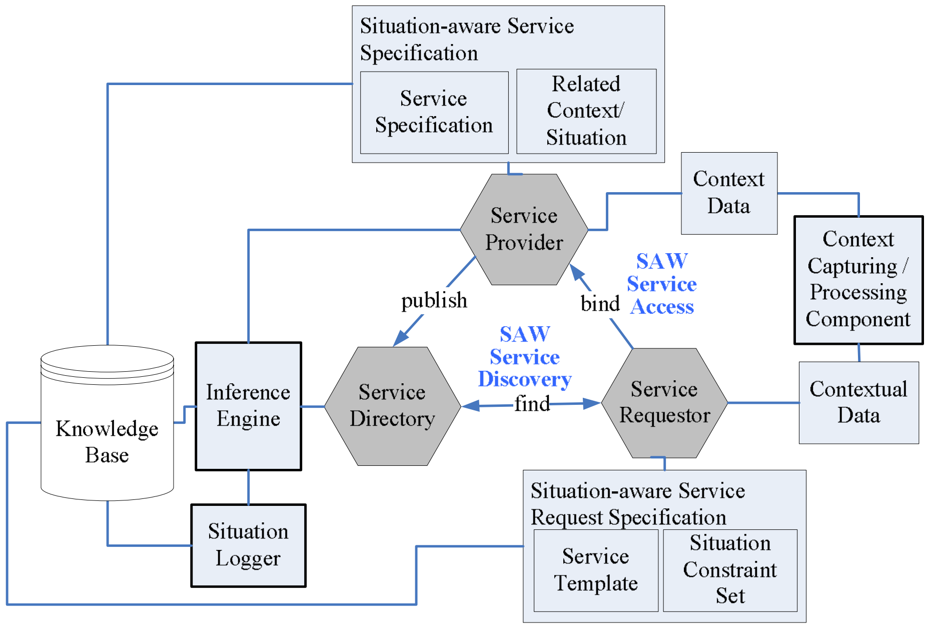

Figure 3. System diagram of SAW-OWL-S situation-aware service-based systems

SAW-OWL-S in the system is used to describe the service specifications together with related contextual data and situation information. A knowledge base stores the specifications, and an inference engine is used to perform logic inferences on the specifications. The system is built on top of the context capturing/processing components, such as Context Toolkit [13]. There is also a situationLogger component associated with the knowledge base and inference engine which will keep tracking of the context/situation change, so that temporal situation evaluation can be performed. As many service-based systems are distributed and may not have central control, the system shown in Figure 3 can have different variations. For example, the service directory can be centralized or distributed. Or there may be no service directory in the system, and the service providers and requestors work in a peer-to-peer mode. Due to the limitation of computation power, the inference engine and situationLogger do not need to be run at each device. They can be deployed only at powerful nodes, like gateways, and other devices can interact with them through network interfaces.

6.2 Logic inferences over SAW-OWL-S specifications

Various formal logic inferences can be performed on SAW-OWL-S specifications for validation and reasoning. They can be categorized to two types:

1. OWL ontology reasoning

SAW-OWL-S is based on OWL DL, which is equivalent to Description Logic (DL). Automated reasoning over the ontology can be performed using a DL Reasoner, such as RACER [14]. Some of the inferences include consistency check, subsumption reasoning and implicit knowledge inference. Consistency check determines whether a service specification is consistent by reasoning if there is any inconsistent ontology class or inconsistent ontology instance in the specification. Subsumption reasoning checks whether an ontology class subsumes another,, and this information can be used to obtain all the implicit subsume relations in the service specification. Implicit knowledge inference can deduce the implicit knowledge conveyed by the service specification.

2. First-order logic rule-based reasoning

The SAW-OWL-S specification also supports rule-based First Order Logic reasoning. Since DL is a decidable part of FOL, it is possible to convert the DL equivalent – OWL DL ontology to FOL specifications and perform reasoning using FOL theorem provers. The basic ideas of transforming OWL representation to FOL representation is to translate class references to unary predicates, translate properties to binary predicates and translate axioms appropriately [15]. There are some efforts to increase the reasoning capability of OWL like SWRL [16], and to extend OWL to support First Order Logic [17]. FOL rules can be used to inference the situation value [12] and whether a service specification satisfies user's request. For example, the service discovery matchmaker can inference whether the printer service satisfies the *OnSameFloorAsUser* situation or not in the scenario presented in Section 5 by using the rule generated from the definition of *OnSameFloorAsUser* situation:

$$(?service\ hasContextData\ \ ?lo) \wedge (?lo\ present\ Location)$$
$$\wedge (?lo\ hasValue\ \ ?v) \wedge (?v\ sameFloor\ userLocation)$$
$$\Rightarrow (?service\ satisfies\ OnSameFloorAsUser)$$

6.3 Interactions in SAW-OWL-S Situation-aware Service-based Systems

With the incorporation of context and situation information in services, the interactions among the service provider, service requestor and service directory can be extended to support SAW. The major interactions in situation-aware service-based systems are described as follows:

I1. Service advertisement: The service provider advertises its specification in the service directory. The service specification includes the traditional OWL-S service profile, service model and service grounding information. With the extended SAW specification capability of SAW-OWL-S, the service can also specify related contextual data and external pre-condition/post-condition situations. It can also specify in which situation it will be triggered. The service directory verifies the consistency of the specification and stores it in the knowledge base.

I2. Service request: The service requestor queries the service directory for a desired service. The service discovery request mainly includes two parts: a service template and a situation constraint set. The service template gives a description of the desired service. It may be as simple as some keywords or as complicated as a full SAW-OWL-S service specification. The situation constraint set includes a set of situations and is used to filter out the unwanted services which fail to satisfy these situation constraints.

I3. Situation-aware service matchmaking: Upon receiving a service discovery request, the service matchmaker is invoked by the service directory. The service matchmaker first uses the situation constraint set to filter out unsuitable services. Then all the suitable service advertisements are matched with the request based on their SAW-OWL-S specifications. A service with greatest similarities is then returned. The situation-aware service matchmaking includes three different levels of matching: 1) Syntax-based matching as used by most of the traditional service matchmaking approaches, such as UDDI [18], which matches services using keyword searching on profile fields; 2) Capability-based matchmaking, which is based on the semantic similarities among ontology concepts, such as the capability-based service matchmaking approach [19]; and 3) Condition-based matchmaking, which matches services based on their pre-conditions and post-conditions. Although the condition-based matching of software components has been studied [20], there is no service matchmaking approach based on conditions due to the difficulty to specify external conditions of the services. With SAW-OWL-S, the external pre-condition and postcondition of a service can be both easily specified using situations. The subsumption relations among different situations can be deduced using formal logic inference. Hence, condition-based service matchmaking is feasible for SAW-OWL-S situation-aware service-based systems.

I4. Service request reply: Service directory replies the service requestor with the best matched service.

I5. Situation-aware service access: The service requestor accesses the discovered service using the provided grounding information. The service provider can verify whether the service requestor satisfies its situation pre-conditions, including the requirements for security and privacy considerations.

I6. Service execution: After the execution of a service, certain external post-condition will be satisfied.

I7. Service composition: Multiple services may constitute a workflow to achieve user objectives dynamically.

I8. Agent discovery: In SOA, a Web service is viewed as an abstract notion that must be implemented by a concrete agent. The agent is the concrete entity (a piece of software) that sends and receives messages, while the service is the abstract set of functionality that is provided [1]. SAW-OWL-S can also be used to describe the functionality provided by a concrete agent and the abovementioned service discovery approach can also be used for agent discovery. This is useful for finding and deploying a suitable agent to perform certain functionality.

7. Conclusions and Future Work

In this paper, we have presented an approach to incorporate situation-awareness in service specifications for situation-aware service-based systems using SAW-OWL-S, an extension of OWL-S with situation ontology. We have also presented a situation-aware service-based system diagram based on SAW-OWL-S and the detailed interactions in the system, including three different matching levels of situation-aware service matchmaking.

Further research which needs to be done in this area includes improving SAW-OWL-S to incorporate temporal logic, improving the performance of logic reasoning over SAW-OWL-S specifications, and establishing algorithms for situation-aware service matchmaking for trustworthy service discovery and situation-aware service access control.

Acknowledgment

The work reported here is supported by the National Science Foundation under grant number ITR-CYBERTRUST 0430565. The authors would like to thank Dazhi Huang of Arizona State University for many valuable discussions.

References

[1] W3C, “Web Services Architecture,” http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.
[2] S. S. Yau, Y. Wang, and F. Karim, "Development of Situation-Aware Application Software for Ubiquitous Computing Environments", *Proc. 26th Ann. Int'l Computer Software and Applications Conf. (COMPSAC 2002)*, 2002, pp. 233-238.
[3] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S.Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing", *IEEE Pervasive Computing,* 1(3), July-September 2002, pp.33-40.
[4] OWL-S: Semantic Markup for Web Services, Available at: http://www.w3.org/Submission/OWL-S/.
[5] W3C, “Web Services Description Language (WSDL) 1.1,” http://www.w3.org/TR/wsdl.
[6] T. Strang, C. Linnhoff-Popien, and K. Frank, “CoOL: A Context Ontology Language to enable Contextual Interoperability”, *Proc. 4th IFIP Int’l Conf. on Distributed Applications and Interoperable Systems (DAIS2003)*, 2003, pp. 236-247.
[7] Soraya Kouadri Mostefaoui, Hannes Gassert and Béat Hirsbrunner, “Context Meets Web Services: Enhancing WSDL with Context-Aware Features”, *1st Int’l Workshop on Best Practices and Methodologies in Service-Oriented Architectures*, 2004. pp. 1-14.
[8] M. Chapman et al. D. Bunting. Web Services Context Service Specification, Ver. 1.0. http://developers.sun.com/ techtopics/webservices/wscaf/wsctx.pdf , 2003.
[9] C. Lee and S. Helal, “Context Attributes: An Approach to Enable Context-awareness for Service Discovery”, *Proc. of the Symposium on Applications and the Internet (SAINT)*, 2003, pp. 22--30.
[10] S. S. Yau, D. Huang, H. Gong and H. Davulcu, “Situation-Awareness for Adaptable Service Coordination in Service-based Systems”, *Proc. 29th Ann. Int'l Computer Software and Application Conf. (COMPSAC 2005)*, 2005, pp. 107-112.
[11] S. S. Yau, D. Huang, H. Gong, and Y. Yao, "Support for Situation-Awareness in Trustworthy Ubiquitous Computing Application Software", *Jour. Software Practice and Engineering*, to appear.
[12] S. S. Yau and Junwei Liu, “Hierarchical Situation Modeling and Reasoning for Pervasive Computing”, *Proc. 3rd Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS2006)*, to appear.
[13] A. K. Dey, “Providing Architectural Support for Building Context-Aware Applications”, PhD thesis, College of Computing, Georgia Institute of Technology, 2000.
[14] V. Haarslev and R. Möller. “Racer: A Core Inference Engine for the Semantic Web”, *Proc.2nd Int’l Workshop on Evaluation of Ontology-based Tools (EON2003)*, October, 2003, pp. 27-36.
[15] Using a First Order Logic Prover with OWL. Available at: http://wonderweb.man.ac.uk/owl/first-order.shtml
[16] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Available at: http://www.w3.org/Submission/SWRL/.
[17] A Proposal for a SWRL Extension towards First-Order Logic. Available at: http://www.w3.org/Submission/SWRL-FOL/.
[18] “Universal Description Discovery and Integration Platform”, Available at: http://www.uddi.org/pubs/ Iru_UDDI_Technical_White_Paper.pdf , September 2000.
[19] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, “Semantic Matching of Web Services Capabilities”, *Proc. of the 1st Int’l Semantic Web Conf. (ISWC2002)*, 2002, pp. 333-347.
[20] A. Moormann Zaremski, and J. M. Wing, “Specification matching of Software Component”, *ACM Trans. on Software Engineering and Methodology*, 1997, pp. 333-369.