QoS based Semi-Automatic Web Service Composition using Multi-Agents Systems

G. Vadivelou[#1], E. Ilavarasan[*2], M.S. Yasmeen[#3]

[#]*Bharathiar University,Coimbatore, Tamilnadu, India*

[*]*Department of Computer Science & Engineering,Pondicherry Engineering College, Pondicherry, India*

[#]*Department of Computer Science,K.M.Centre for P.G. Studies, Lawspet, Pondicherry, India*

Abstract-This paper presents an agent and ontology based approach that supports the semi-automatic composition of Web services. A Web service is an accessible application that other applications and humans can discover and invoke to satisfy multiple needs. To reduce the complexity featuring the composition of Web services, two concepts are put forward, namely, software agent and ontology. An agent is an entity that acts on behalf of others in an autonomous fashion, performs its actions in some level of pro-activity and reactivity and exhibits some levels of the key attributes of learning, co-operation, and mobility. Agent Based Systems (ABS)[11] may be divided, roughly, into individual agents, and multi-agent systems (MAS)[11]. Agent technology has been a hot topic, and most likely, this is mainly due to the popularity of the Java programming language, which represents an ideal language for implementing software agents as it is the "Write Once Run Anywhere" language. Ontology is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain. The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies. This paper provides the way to select an optimal composition of services and it also propose a framework for Semi-Automatic Web Services Composition.

Keywords- Agents, Web Service, Web Service Composition , OWL-S, WSIG

I. INTRODUCTION

Web Services[1] are considered as self-contained, self describing, modular applications that can be published, located, and invoked across the Web. Amount of products and services available now on the Web increases dramatically and goes beyond user's ability to analyse them efficiently. At the same time the number of potential customers available via the Internet also increases significantly and starts to be beyond service providers' ability to perform efficient targeted marketing. Another important issue related to the development of Web services is their integration and composition. Recent progress in the field of Web Services has made it practically possible to publish, locate, and invoke applications across the Web. This is a reason why more and more companies and organizations now implement their core business and outsource other application services over the Internet. In particular, if no single Web service can satisfy the functionality required by a user, there should be a possibility to combine existing services together in order to fulfill the request. The challenge is that Web services can be created and updated on the fly and it is often beyond human capabilities to analyse the required services and compose them manually.

The remainder of this paper is organized as follows. In the next section, we will introduce the basic concepts such as Web Service, Agents, Web Service Composition and OWL-S. Section III describes our related work. Section IV discusses about proposed framework. Section V discusses about the relationship between Agent and Ontology and finally, the paper concludes with the future work in Section VI.

II. BACKGROUND

A . Web Service

A Web Service [4] is an accessible application that other applications and humans as well, can automatically discover and invoke. An application is a Web Service if it is

(1) independent as much as possible from specific platforms and computing paradigms;
(2) Developed mainly for inter organizational situations rather than for intra-organizational situations; and
(3) Easily composable (i.e., its composition with other Web services does not require the development of complex adapters)Web Services are, in practice, transient and stateless processes that exist only during service execution, which is triggered by a request coming from a consumer, or client. Services are instantiated to perform specific tasks, thus facilitating scalable, concurrent service provision. The design of a Web Service is usually defined as a clearly articulated workflow, for the sake of reliability and quality of service.

Though Web Services has many advantages, but still there are certain problems which need to be addressed. These are:
(1) Provided resources and services are not in machine understandable form, these are in human understandable form
(2) The representation of resources and services on the web are unstructured and they are loosely related to each other
(3) Searching resources and services on the web at present is keyword based; no semantics of the resources are used. So by using some popular keywords, web page owner can make his page mostly retrieval with irrelevant results and
(4) Interoperability between toolkits.

B. Semantic Web Services

Semantic web technology has drawn a considerable attention of the researchers in the field of distributed information systems, artificial intelligence, and so on. Researchers are taking interest to make use of semantic web technology as a central component of their software constructions. The web services are lacking the semantic description, the semantic web researchers have proposed to augment web services with a semantic description of their functionality in order to facilitate their discovery and integration. This technology, combination of web services with semantic web technology, is referred as semantic web services (SWS). SWS is, therefore, an extension of web service with an explicit representation of meanings. SWS will support the automatic discovery, composition, and execution of web services. Hence, it has the potentiality to alter the way knowledge and business services are provided and used on the web.

C. Agents

An Agent is a piece of software that acts autonomously to undertake tasks on behalf of users. The design of many Agents is based on the approach that the user only needs to specify a high-level goal instead of issuing explicit instructions, leaving the how and when decisions should be taken by the agent. An software agent (SA)[10] exhibits a number of features that make it different from other traditional components including autonomy, goal orientation, collaboration, edibility, self-starting, temporal continuity, character, communication, adaptation, and mobility.

An agent is an entity that:

- Acts on behalf of others in an autonomous fashion
- Performs its actions in some level of pro-activity and reactivity
- Exhibits some levels of the key attributes of learning, co-operation, and mobility.

Software agents are an innovative technology designed to support the development of complex, distributed, and heterogeneous information systems. There is however no complete standard/consensus definition of an agent. As a result, agents tend to be characterized in terms of a number of their behavioural attributes.

1) Autonomy: the ability to act autonomously to some degree on behalf of users for example by monitoring events and changes within their environment.
2) Pro-activity: the ability to pursue their own individual set goals, including by making decisions.
3) Re-activity: the ability to react to and evaluate external events and consequently adapt their behaviour and make appropriate decisions to carry out the tasks to help them achieve their goals.
4) Communication and Co-operation: the ability to behave socially, to interact and communicate with other agents (in multiple agent systems (MAS)) i.e. exchange information, receive instructions and give responses and co-operate when it helps them fulfil their own goals.
5) Negotiation: the ability to conduct organized conversations to achieve a degree of co-operation with other agents.
6) Learning: the ability to improve performance over time when interacting with the environment in which they are embedded.

For our research purposes, we further characterize a software agent as a running program object, capable to initiate, receive, execute or reject a message autonomously to attain its goals during its life cycle.

D. Java Agent Development Environment (JADE)

JADE is fully developed in Java and is based of the following driving principles:
• Interoperability – JADE is compliant with the FIPA specifications. As a consequence, JADE agents can interoperate with other agents, provided that they comply with the same standard.
• Uniformity and portability – JADE provides a homogeneous set of APIs that are independent from the underlying network and Java version. More in details, the JADE run-time provides the same APIs both for the J2EE, J2SE and J2ME environment. In theory, application developers could decide the Java run-time environment at deploy-time.
• Easy to use – The complexity of the middleware is hidden behind a simple and intuitive set of APIs.
• Pay-as-you-go philosophy – Programmers do not need to use all the features provided by the middleware. Features that are not used do not require programmers to know anything about them, neither add any computational overhead

E. Web Service Integration Gateway (WSIG)

The objective of WSIG is to expose services provided by agents and published in the JADE DF as web services with no or minimal additional effort, though giving developers enough flexibility to meet specific requirement. The process involves the generation of a suitable WSDL for each service-description registered with the DF and possibly the publication of the exposed services in a UDDI registry.

The WSIG add-on supports the standard Web services stack, consisting of WSDL for service descriptions, SOAP message transport and a UDDI repository for publishing Web services using tModels. As depicted in **Error! Reference source not found.**. WSIG is a web application composed of two main elements:

- WSIG Servlet
- WSIG Agent

The WSIG Servlet is the front-end towards the internet world and is responsible for

- Serving incoming HTTP/SOAP requests
- Extracting the SOAP message
- Preparing the corresponding agent action and passing it to the WSIG Agent

Moreover once the action has been served

- Converting the action result into a SOAP message
- Preparing the HTTP/SOAP response to be sent back to the client

The WSIG Agent is the gateway between the Web and the Agent worlds and is responsible for

- Forwarding agent actions received from the WSIG Servlet to the agents actually able to serve them and getting back responses.
- Subscribing to the JADE DF to receive notifications about agent registrations/de registrations.

- Creating the WSDL corresponding to each agent service registered with the DF and publish the service in a UDDI registry if needed.

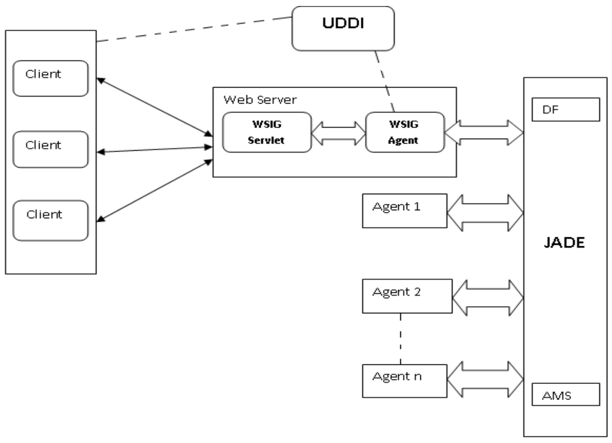


Fig. 1 WSIG Architecture

Two main processes are continuously active in the WSIG web application:

- The process responsible for intercepting DF registrations/deregistration and converting them into suitable WSDLs. As mentioned, this process is completely carried out by the WSIG Agent.

- The process responsible for serving incoming web service requests and triggering the corresponding agent actions. This process is carried out jointly by the WSIG Servlet (performing the necessary translations) and the WSIG Agent (forwarding requests to agents able to serve them).

F. Overview of Ontology and OWL-S

Ontology's are widely used in various fields like Semantic Web, Artificial intelligence as form of Knowledge representation. Protégé is an Ontology Editor that allows you to design and query ontologies. Bean generator plug-in generates java code for the ontology and it also uses these objects in messages.

The principle objectives of OWL-S are:

(1) to provide a general-purpose representational framework in which to describe Web Services
(2) to support automation of service management and use by software agents
(3) to build, in an integral fashion, on existing Web Service standards and existing Semantic Web standards; and
(4) to be comprehensive enough to support the entire lifecycle of service tasks.

OWL-S is an OWL ontology[17] that may be used to specify semantically rich characterizations of services on the Web. OWL-S is organized into four parts.

The *profile* describes capabilities and discriminating features of Web services for purposes of advertising and matchmaking.

The *process model* provides a description of the structure of activities involved in providing the service, from which service requesters can derive information about service invocation and interaction patterns.

The *grounding* is a description of how abstract information exchanges described in the process model are mapped onto actual concrete messages that flow between the provider and the requester.

Finally, the *service* itself provides a means of bundling together instances of the top-level profile, process, and grounding classes that are meant to be used together. OWL-S complements industry efforts such as SOAP, WSDL[5] and BPEL4WS [18]. It builds upon these efforts by adding rich typing and class information that can be used to describe and constrain the range of Web service capabilities more effectively than can be done with XML data types. Further, in the process model, it captures not only the control flow and data flow of Web services, but also their prerequisites and side effects (preconditions and effects) in the world. OWL-S' basis in OWL[30] enables the grouping of like services and like data types into taxonomic hierarchies, together with definitions of the relationships and constraints between classes and their instances. The well-defined semantics enables formal automated manipulation of these structures, with known outcomes, thus providing a foundation for automation of a variety of Web service operations, such as discovery, matchmaking, interoperation, composition, enactment, monitoring, and recovery.

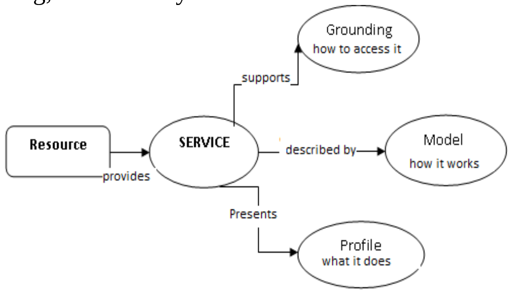


Fig. 2 Overview of OWL-S

G. Inputs, outputs, preconditions, and results

Understanding all three components of an OWL-S service model[10] requires understanding inputs, outputs, preconditions, and results. The inputs are the object descriptions that the service works on; the outputs are the object descriptions that it produces. These descriptions are of course not known when the process is defined; all we can expect is the specification of their types (as OWL classes, at the most abstract level). A precondition is a proposition that must be true in order for the service to operate effectively. Results consist of effect and output specifications. An effect is a proposition that will become true when the service completes. In general, these propositions are not statements about the values of the inputs and outputs, but about the entities referred to in the inputs and the outputs.

III. RELATED WORK

Service composition[23] has been the subject of many research projects, such as the Ninja project and SAHARA which includes specifications for WSDL[5], SOAP and other protocols that may be used to describe, access, execute, and discover services on the Web. There are several works on incorporating agents into Web Service systems.

In particular, Gibbins [2] et al demonstrated usage of DAML-S for Web Services descriptions within agents. Another step towards incorporating Web Services into agents is proposed by Ardissono et al . Since their focus has been set to non-symbolic negotiation, their work could be seen as a complementary part to our work, where we focus on logic-based Web Services Composition.

Zakaria Maamar [9] develops a service composition framework, in which multiple-agent-system that composes of composition agent, service agent and service instance agent is the engine of service composition. During the composition process, software agents engage in conversations with their peers to agree on the Web Services that participate in this process. Conversations between agents take into account the execution context of the Web Services. But this paper doesn't consider context aware service.

In this paper, Korhonen, et al. describes the creation of a workflow ontology that is used to describe both agents and Web services. They hope to build a workflow enactment mechanism that can utilize the ontology to bridge the communications gap between agents and Web Services.

David Martin[10] discusses how to use the OWL-S. This paper shows how to use OWL-S in conjunction with Web service standards, and explains and illustrates the value added by the semantics expressed in OWL-S. But this paper doesn't consider Web service composition.

IV. PROPOSED FRAMEWORK

The proposed framework (See Figure 3) supports the construction and execution of semi-automatic service composition. The system architecture is based on three categories of components: Service Discovery Component[26], Process Building Components, and Process Configuration & Execution Components

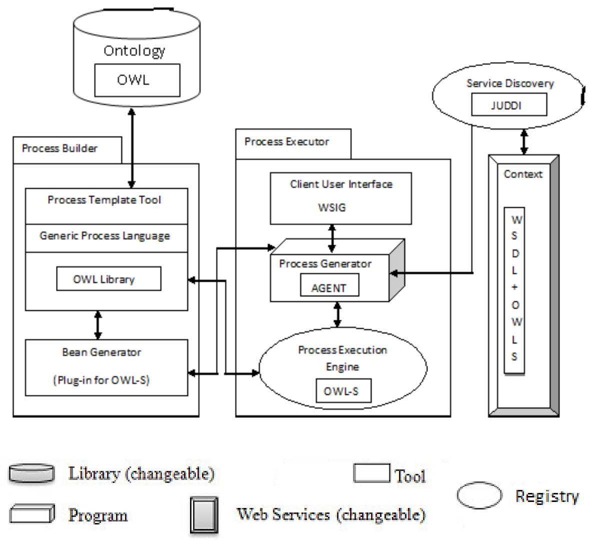
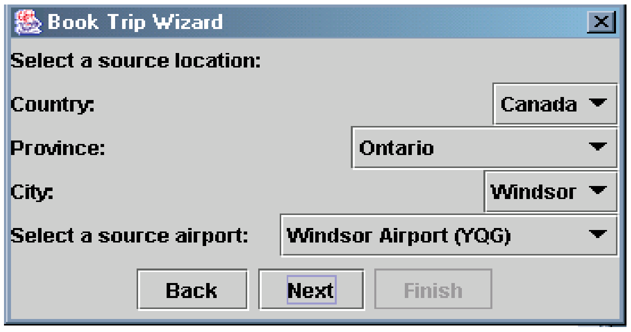


Figure 3: Proposed Framework for Semi-Automatic Composition

1) Service Discovery Component
The providers publish their web services on a web services registry.
[Service Discovery]: The Service Discovery & Registry has registry, discovery and selection functions. The web services are registered in JUDDI registry and then the web services uses a gateway WSIG to integrate the requested services by comparing their semantic descriptions with the available registered services.
2) *Process Building Components*
The process developer uses a IDE in order to build a generic process template. It uses a published domain ontology which is related to a specific organization to describe participating activities semantically. Protégé is an Ontology Editor that allows you to design and query ontologies. Bean generator plug-in generates java code for the ontology and it also uses these objects in messages
3) *Process Configuration and ExecutionComponents*
The Client uses the Agent programming and WSIG to configure a process template and to compose the optimal web services. Then the process can be executed. The following components allow realizing this objective:
[Client User Interface]: a add-ons which handles the communication between the end-user and the platform. It lets the user choose a optimal web services in order to achieve the composition. After process configuration, the process is executed .
[Process Generator]: Agent handles the process configuration and converts the generic process into an executable one.
[Execution Engine]: OWL-S is an ontology and a language to describe web services The project aims to create an easy-to-use editor for creating OWL-S services. The editor is being developed as a plug-in to the protégé ontology editor. Some important features of OWL-S editor are: good overview; graphical editing; import/export; WSDL support; and input/output/precondition/ result manager.

V. IMPLEMENTATION

Travel Agency Management System (TAMS) is taken as the case study The purpose of this case study is to create an agent-based software system which gives the customers about the travelling details and also gives information about flights, rail and accommodation. The case study is implemented in Java1.6 using JADE[15] framework for Agent programming, Protégé for creating ontologies[17] ,OWL-S[10] editor for composing the web services and WSIG 2.3 for integrating agents and web services. Sample screen shots are shown in the figures 4,5,6,7,8 and 9.


Book Trip Wizard

Select a source location:

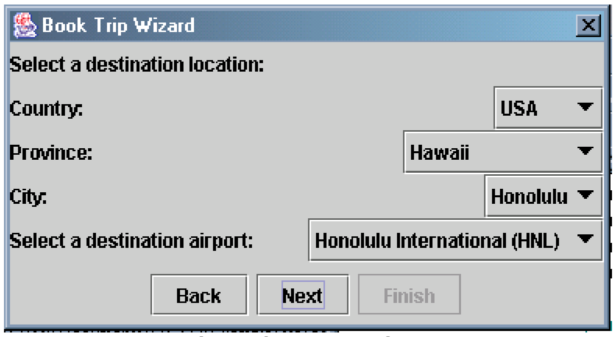
Country: Canada

Province: Ontario

City: Windsor

Select a source airport: Windsor Airport (YQG)

Figure 4: Selecting the Source Location In TAMS



Select a destination location:

Country:	USA
Province:	Hawaii
City:	Honolulu
Select a destination airport:	Honolulu International (HNL)

Figure 5: Selecting the Destination location in TAMS

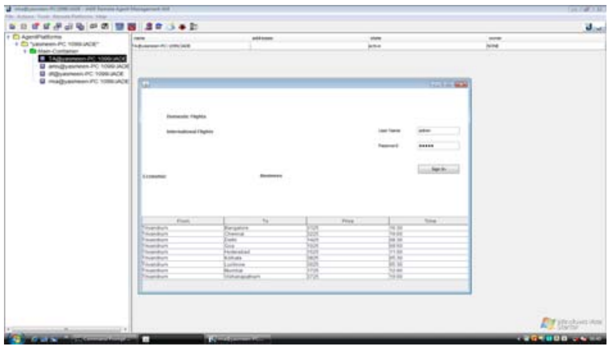


Figure 6: Agent Communication in JADE

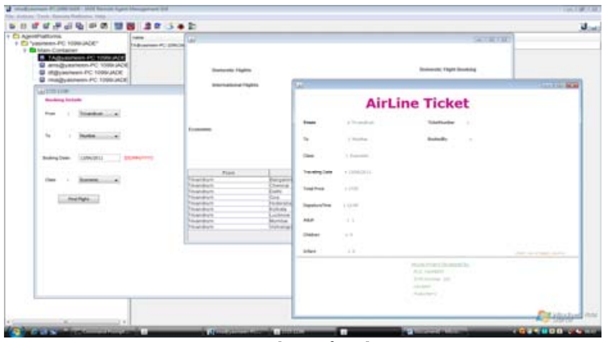
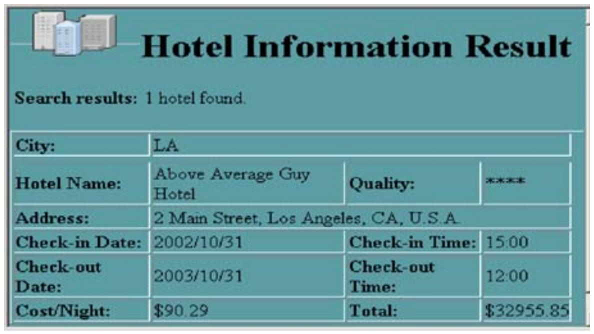


Figure 7: Screenshots of Airline Reservation



Hotel Information Result

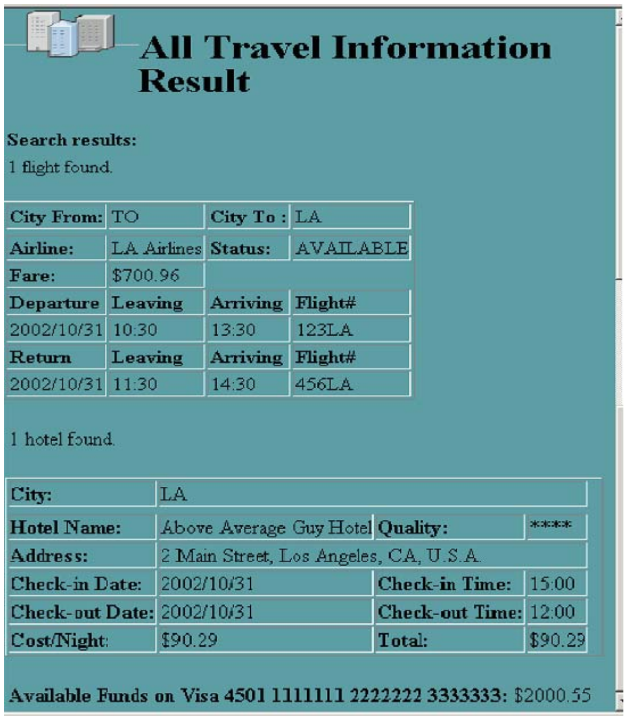
Search results: 1 hotel found.

City:	LA		
Hotel Name:	Above Average Guy Hotel	Quality:	****
Address:	2 Main Street, Los Angeles, CA, U.S.A.		
Check-in Date:	2002/10/31	Check-in Time:	15:00
Check-out Date:	2003/10/31	Check-out Time:	12:00
Cost/Night:	$90.29	Total:	$32955.85

Figure 8: Screenshots of Hotel Reservation

A) Integrating Agent and OWL-S

OWL-S[10] explicitly supports the description of services as classes of activities, so that agents can reason about the possible benefit of using them, determine the content of the messages necessary to invoke them, and interpret responses from them. This is substantially different than the rationale behind *agent communication languages* (ACLs), such as the Knowledge Query and Manipulation Language (KQML) or the Foundation for Intelligent Physical Agents (FIPA) ACL, developed during the 1990's. ACLs like KQML and FIPA [22] were designed primarily to provide a uniform syntax and semantics for messages with arbitrary content, passed between software agent peers. In contrast, the OWL-S process ontology is a framework for describing more abstractly the service activities themselves, and likely sequences of such activities. OWL-S[14] descriptions of the inputs and outputs of individual atomic activities characterize the information conveyed in the underlying WSDL input and output messages. The OWL-S grounding model translates the inputs and outputs of OWL-S service descriptions from OWL[12] into the XML elements of corresponding WSDL messages. But it can just as properly be used to translate that information into a KQML or FIPA message format for communications with agents that provided services using an ACL message transport mechanism.



All Travel Information Result

Search results:

1 flight found.

City From:	TO	City To :	LA
Airline:	LA Airlines	Status:	AVAILABLE
Fare:	$700.96		
Departure	Leaving	Arriving	Flight#
2002/10/31	10:30	13:30	123LA
Return	Leaving	Arriving	Flight#
2002/10/31	11:30	14:30	456LA

1 hotel found.

City:	LA		
Hotel Name:	Above Average Guy Hotel	Quality:	****
Address:	2 Main Street, Los Angeles, CA, U.S.A.		
Check-in Date:	2002/10/31	Check-in Time:	15:00
Check-out Date:	2002/10/31	Check-out Time:	12:00
Cost/Night:	$90.29	Total:	$90.29

Available Funds on Visa 4501 1111111 2222222 3333333: $2000.55

Figure 9: Overall Screenshots of TAMS

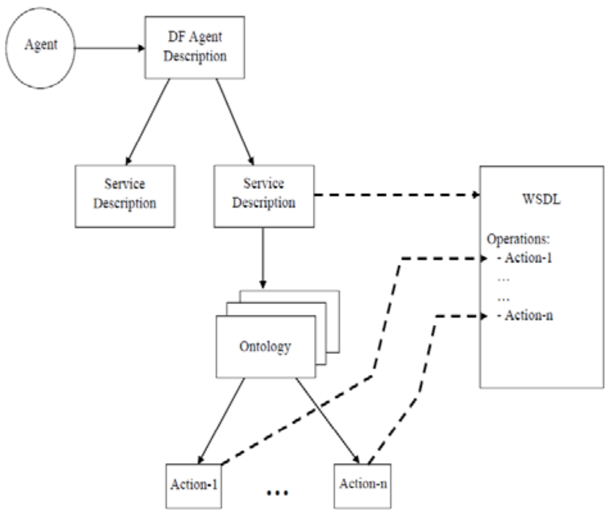


Figure 10: Agent and Ontology Architecture

OWL-S groundings[29] used with KQML or FIPA[22] must relate atomic processes to ACL message patterns with specific performatives and perhaps even specific content forms. The performatives referenced in these messages patterns depend on the type of service provided, and the kind of action triggered by the message. In summary, OWL-S abstracts[27] away the details of the message-level interactions of both ACLs and web service message languages like WSDL. Instead, it focuses on characterizing the content and workflow of interactions with services so that client systems can perform the reasoning necessary to interoperate with them automatically. The ontology and agent architecture is shown in Figure 10.

VI. CONCLUSIONS AND FUTURE ENHANCEMENTS

This paper has proposed an framework for semi-automatic composition at abstract service using agents, web services and ontology. In future enhancement, the Web service composition[7] can be done in Automatic Methods[19] using AI Techniques considering QoS factors of web services.

REFERENCES

[1] W3C. Web services activity (web site)". http://www.w3c.org/2002/ws/

[2] N.Gibbins, S. Haris and N.Shadbolt. "Agent- Based Semantic Web Services". In Proceedings of the 12th Int. WWW Conf., WWW2003, Budapest, Hungary, 2003, ACM Press, 2003, pp.710-717.

[3] S. McIlraith, and T.C. Son. "Adapting Golog for composition of Semantic Web services", *Knowledge Representation and Reasoning (KR2002)*, Toulouse, France, 2002.

[4] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana, "The Next Step in Web Services," Comm. ACM, vol. 46, no. 10, Oct. 2003.

[5] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1, 2001. At http://www.w3.org/TR/2001/NOTE-wsdl- 20010315

[6] J. Rao, P. Kungas, and M. Matskin, "Logic-based Web services composition: from service description to process model", *The 2004 Intl Conf on Web Services*, San Diego, USA, 2004.

[7] S. McIlraith, T. Son, and H. Zeng. "Semantic web services", *IEEE Intelligent Systems*, 2001, 16(2):46–53.

[8] T. Andrews et. al., BPEL v1.1. (2007). [Online].Available:http://www.ibm.com/developerworks/library/specification/ws-bpel/

[9] Zakaria Maamar, Soraya kouadri Mostefaoui and Hamdi Yahyaoui, "Towards an Agent Based and Content Oriented Approach for Web Services Composition", IEEE Transactions on knowledge and Data Engineering, 2005.

[10] David Martin, Mark Burstein, Sheila Mc Ilraith, Massimo Paoulucci and Katia Sycara, "OWL-S and Agent Based Systems", Members of the OWL-S Coaliation", USA, 2004.

[11] David Martin, Massimo et al, "Bringing Semantics to web services: The OWL-S approach", Artificial Intelligence Center, SRI International, Menlo Park CA, USA, 2004.

[12] Jing Dong, Yongtao sun and Sheng Yang, "OWL-S Ontology Framework Extension for Dynamic web service composition", USA, 2006.

[13] Zhonghua Yang, Zhang Jing Bing, Jiao Tao, Robert Gay, "Characterizing services composeability and OWL-S based services composition", Singapore, 2005.

[14] David Martin et al, "Bringing Semantic to web services with OWL-S", Springer science and Business Media, LLC, 2007.

[15] Ahmed Sallam, Zhiyong Li and Shaimea Hassan, "Web services supervision system based on JADE", China, JDCTA journal, 2010.

[16] Farhan Hassan Khan et al, "QoS based dynamic web composition and execution", In proceeding of Int'l journal of computer Science and Information Security(IJCSIS), 2010.

[17]Jose Luis Ambite et al, "Argos: An ontology and Web service composition Infrastructure for goods movement Analysis", USA, 2004.

[18] Muhammad Ahtisham Aslam et al, "From BPEL4WS process model to Full OWL-S ontology", School of Computer and Information sciences, University of South Australia, Australia, 2006

[19] Mithun Sheshagiri, "Automatic Composition and Invocation of Semantic web services", M.Sc thesis, Department of Computer Science, University of Maryland, 2004.

[20] Lin Padgham and Michael winikoff, "Developing Intelligent Agent Systems, A practical guide", RMIT University, Melbourne, Australia, 2005.

[21] Jing Dong et al, "Dynamic Web service composition based on OWL-S", Springer, 2006.

[22] Esteban Leon Soto, "FIPA Agents Messaging grounded on web services", German Research center for Artificial Intelligence (DFKI), 2006.

[23] Biplav srivastava and Jana Kochler, "Web service composition- current solution and open Problems", ICAPS, 2003.

[24]R.Jayaprakash and R. vimal Raja, " Evaluating web service composition methods with the help of a Business Application", In proceeding of the Int'l journal of Engineering Science and Technology(IJEST), IEEE, 2010.

[25] Jinghai Rao and Xiaomeng su, "A survey of automated web service composition methods", Department of Computer and Information Science, Norway, 2005.

[26] Abdaladhem Albreshne and Jacques Pasquier, "Semantic-based Semi-automatic Web service composition", Computer Department, Switzerland, 2010.

[27] Katia Sycara and David Martin, "Tools and Technologies for semantic web services: An OWL-S perspective", ISWC, 2006.

[28] N. Kavantzas, D. Burdett, and G. Ritzinger, WSCDL v1.0. (2004). [Online]. Available:
http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/

[29] OWL-S: Semantic Markup for Web Services, W3C Member Submission, (22 November 2004). [Online].
Available: http://www.w3.org/Submission/OWL-S/.

[30] Deborah L. McGuinness and Frank van Harmelen, "OWL Web Ontology Language Overview". World Wide Web Consortium (W3C) Candidate Recommendation. August 18, 2003. At http://www.w3.org/TR/owl-features/

[31] The Universal Description, Discovery and Integration (UDDI) protocol. Version 3, 2003. At http://www.uddi.org/