Engineering an MAS Platform for Semantic Service Integration based on the SWSA

Özgür Gümüs[1], Önder Gürcan[1], Geylani Kardas[2],
Erdem Eser Ekinci[1], and Oguz Dikenelli[1]

[1]Ege University, Department of Computer Engineering, 35100 Bornova, Izmir, Turkey
`{onder.gurcan,ozgur.gumus,oguz.dikenelli}@ege.edu.tr`
`erdemeserekinci@gmail.com`
[2]Ege University, International Computer Institute, 35100 Bornova, Izmir, Turkey
`geylani.kardas@ege.edu.tr`

Abstract. In this paper, a Multi-Agent System (MAS) platform for semantic service integration based on the Semantic Web Services Initiative Architecture (SWSA) is discussed. We define a software architecture in order to provide concrete realization of the SWSA. The architecture fullfills fundamental requirements of the SWSA's sub-processes. Software agents are employed in automatic discovery and execution of the Semantic Web Services within this architecture. We also elaborate implementation of SWSA's sub-processes (service advertisement, discovery, engagement and enactment) taking the main components of the defined architecture and their interactions into consideration. Hence, the developers can easily utilize semantic web service technologies by using this flexible and extensible platform.

1 Introduction

Web services enable the software components on different platforms to interact with others conforming some specific description and communication standards. So, they are supported strongly by industrial players in the Internet computing area. On the other hand, software agents are entities that perform actions to achieve user's goals by interacting with other agents. Any software agent can use existent web services dynamically/automatically to perform an action which is necessary for achieving its user's goals. However, they use different communication and coordination standards from web services and they need some semantic knowledge about these services to reason in order to use them dynamically. At this point, the semantic web service concept can help us since semantic web services are web services whose functionalities and execution details are described using ontologies. However, there are still some problems and uncertain situations to succeed this cooperation.

The Semantic Web Services Initiative[1] Architecture (SWSA) committee[2] has created a set of architectural and protocol abstractions that serve as a foundation for semantic web service technologies [1]. The proposed SWSA framework builds on the

[1] http://www.swsi.org/, last access on May 16, 2007.
[2] http://www.daml.org/services/swsa/, last access on May 16, 2007.

W3C Web Services Architecture working group recommendation[3] and attempts to address all requirements of semantic service agents: dynamic service discovery, service engagement, service process enactment and management, community support services, and quality of service (QoS). The SWSA framework also determines the actors of each phase, functional requirements of each phase and the required architectural elements to accomplish these requirements in terms of abstract protocols. This architecture is based on the multi-agent system (MAS) infrastructure because the specified requirements can be accomplished with asynchronous interactions based on predefined protocols and using goal oriented software agents.

Although SWSA defines a detailed conceptual model based on MAS infrastructure and semantic web standards, it does not define the software architecture to realize this conceptual model and does not include the theoretical and implementation details of the required software architecture. Hence, in this paper we introduce a completely working subset of SWSA that fulfills fundemental requirements of SWSA's conceptual model. We define a coherent software platform to enable the developers to utilize semantic web service technologies with an engineering perspective. The provided agent platform has the following capabilities which make it flexible and extensible:

- The platform can utilize semantic web technologies to represent and manipulate knowledge and semantic web service technologies to perform its tasks.
- Service provider agent accepts both pure web services and services with a semantic interface. However, admin of this agent must resolve ontology mismatches between platform and service ontology and process mismatches between goal template and service process model using the tools provided by the service provider agent.
- The user creates agent plans which may include instances of predefined goal templates as tasks.
- There are predefined generic plans for each phase of the service execution process: discovery, selection/engagement and enactment/invocation. These plans can be specialized for different service execution needs of application dependent agent plans and they are executed by a special planner [8].

There are some standardization efforts for semantic web services to allow the web services to work in the semantic web environment. The most attractive ones are OWL-S[4] and WSMO[5]. OWL-S is an ontology system for describing web services but it's not a complete system and meaning of some of its elements is not clearly defined. WSMO is said to be more complete framework but it is not based on W3C standards such as OWL and SWRL[6]. Also it does not make use of OWL ontologies and it looks like a workflow system in a distributed and heterogeneous service environment. We preferred OWL-S

[3] W3C Web Services Architecture Working Group, Web Services Architecture Recommendation, 11 February 2004, http://www.w3.org/TR/ws-arch/, last access on May 16, 2007.

[4] Semantic Markup for Web Services, http://www.daml.org/services/owl-s/, last access on May 18, 2007.

[5] Web Service Modelling Ontology, http://www.wsmo.org/, last access on May 18, 2007.

[6] Semantic Web Rule Language, http://www.w3.org/Submission/SWRL/, last access July 18, 2007.

for defining agent's goals and services and external semantic web services but we also implemented the mediation notions mentioned in WSMO.

The paper is organized as follows: in Section 2 our proposed software architecture is discussed. Phases of the semantic service integration in this architecture are explained in Section 3. Evaluation of the architecture within the scope of a real system implementation is given in Section 4. Section 5 contains the related work and finally Section 6 concludes the paper and discusses the future work.

2 Agent Based Semantic Service Architecture

Since the SWSA is conceptual and has a broad perspective, we have some assumptions to implement this architecture in a reasonable way:

– There is a platform ontology which represents the working domain of the platform. This ontology is designed by platform's administrator and stored and managed by platform's ontology agent.
– There are predefined goal templates which the users of the platform may want to achieve by delegating these goals to an agent as a plan. These goal templates are described similar to semantic web service descriptions (inputs, outputs, preconditions and effects) by platform's administrator and stored in a service registry agent.
– Agent services are advertised on a registry agent with their semantic descriptions. These services could be internal capabilities of regular agents or external semantic web services which are included to the platform by service provider agents.
– The mappings between platform's ontology and the ontology that the semantic service depends on must be defined if they are different. Otherwise the service cannot be used by platform's agents.

In order to concretely fulfill fundamental requirements of the aforementioned SWSA's sub-processes, we propose a software architecture in which software agents are employed in automatic discovery and execution of the Semantic Web Services on behalf of their human users (Figure 1).

The architecture ensures utilization of both pure web services and services with the semantic interface via service provider agents. In fact, the architecture presents an IEEE FIPA[7] compliant MAS and member agents of this system also constitute main components of the proposed architecture which are called Service Provider Agent, Service Requester Agent, Service Registry Agent and Ontology Agent. Communication between these agents takes place according to the well known Agent Communication Language (ACL)[8] infrastructure.

Service Provider Agent (SPA) realizes inclusion of pure web services (WSDL) and semantic web services (OWL-S, WSMO etc.) into the MAS and supports agent - semantic service interaction. In case of a web service inclusion, the admin of SPA first

[7] Institution of Electrical and Electronics Engineers (IEEE) Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org/, last access on May 16, 2007.

[8] FIPA Agent Communication Language Specifications, http://www.fipa.org/repository/aclspecs.html, last access on May 16, 2007.

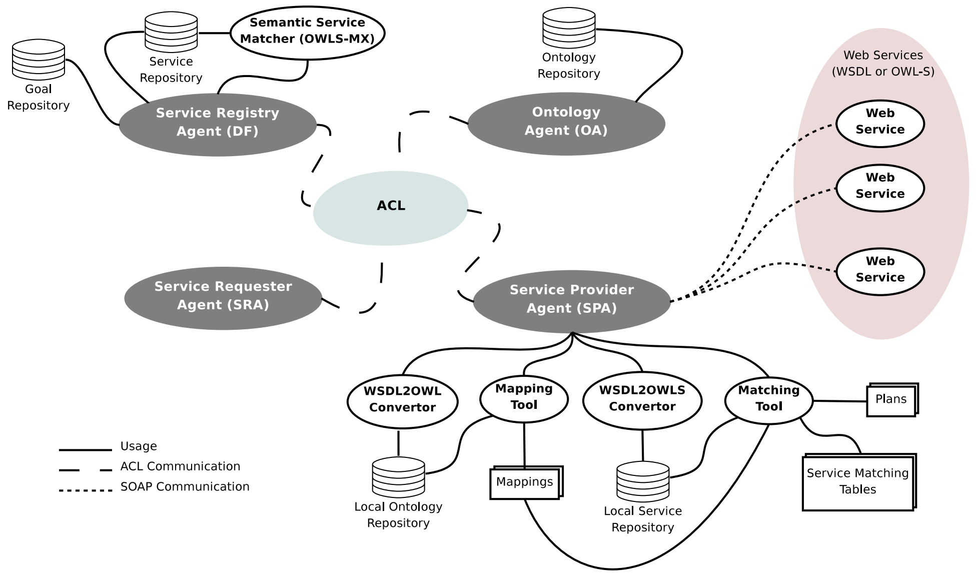

Fig. 1. The Architecture of the MAS Platform for Semantic Service Integration

gives SPA the address of the service description document. Then SPA makes the required mappings and matchings and prepares plans for the service and finally advertises it to the platform as an agent service. In order to perform these operations SPA uses these components: WSDL2OWL Converter, Mapping Tool, WSDL2OWLS Converter and Matching Tool. WSDL2OWL Converter converts the concepts in a given WSDL to OWL concepts. Mapping Tool is a tool which defines mappings between given two ontologies via human interaction. The tool saves mapping knowledge as instances of a mapping ontology for the future use. WSDL2OWLS Converter converts the service description in a given WSDL to an OWL-S service description. A similar service profile generation approach can be found in [11]. Matching Tool helps user to find an appropriate goal template for the service and defines matching between that goal template and the description of the service. Matching Tool also helps to create service specific plans in order to realize process mediation.

Service Requester Agent (SRA) is a service client agent in the architecture. In order to find and execute services, SRA uses the processes which are defined in SWSA. When SRA needs a service it first retrieves a list of semantically appropriate services from Service Registry Agent (*discovery*), then engages with the provider of a suitable service (SPA) (*engagement*) and finally requests SPA to execute the service (*enactment*).

The *Service Registry Agent* is the Directory Facilitator (DF) of our IEEE FIPA compliant MAS. This DF advertises capabilities of the services provided by the agents. It includes a *Service Repository* which stores service advertisements as OWL-S Profiles registered by the corresponding SPAs. It can perform semantic service matching between a requested goal definition and advertised service definitions in order to determine semantically most appropriate OWL-S services for the request. For the capability matching, our DF in the architecture employs a Semantic Service Matcher called OWLS-MX [9]. OWLS-MX is a hybrid semantic web service matcher that utilizes both logic based reasoning and content based information retrieval techniques for services specified in OWL-S. On the other hand, the DF of the architecture also stores agent goal templates of the platform in a repository called *Goal Repository*. Services advertised by the platform agents must conform to these templates stored in this repository. Also agents can use these templates in order to specify their goals.

The *Ontology Agent* (OA) includes an *Ontology Repository* in which ontologies used in the platform are stored. The OA provides controlled access and query on these platform ontologies for other members of the platform.

3 Phases of Semantic Service Integration

The design of the architecture is organized around four main phases for services: (1) the *advertisement phase* in which the inclusion of the web service to the platform is performed by the provider agent; (2) the *discovery phase* that the intended service is explored by the requester; (3) the *engagement phase* in which requester and provider agents make an agreement; (4) the *enactment phase* that the service is invoked by a requester agent via the provider agent.

3.1 Service Advertisement

In this phase external services are included into the platform by SPAs. As mentioned above, agents cannot advertise any service to the platform, unless the service is compatible with the platform's goal templates. SPA accepts both pure web services and services with a semantic interface. Advertisement of these external services is quite similar to the advertisement of agent services. However these services must be semantically adapted to the platform in order to be advertised by SPA. This adaptation involves both data and process adaptations. That is, both the concepts which are used by the service must be suitable to the platform's domain and the work which is done by the service must be compatible with the platform's goal templates.

To advertise an external service, the admin of SPA requests SPA to start the advertisement phase by giving the address of service description document of the service. SPA loads the description and checks whether the service is pure or semantically described. If it is pure, SPA converts the concepts in that service description to semantic concepts using WSDL2OWL converter and stores it as an ontology document (local ontology) in the Local Ontology Repository. Then mappings between this local ontology and platform ontology are defined using Mapping tool with the help of the admin. This process generates a mapping knowledge and it is stored in a file. After this, SPA converts the service description (WSDL) to a semantic service description using WSDL2OWLS converter and stores it in the Local Service Repository. Then SPA tries to adapt this description to a platform's goal template. This is required because agents can only advertise services which are compatible to the platform's goal templates. To do this, SPA requests the goal templates from DF and starts its Matching tool. Matching tools creates a Service Matching Table which holds mapping knowledge, local service profile, the goal template and a plan to enact the service. Finally SPA requests DF to advertise the service using its service description. The AUML[9] sequence diagram for the steps of the external web service advertisement is given in Figure 2.

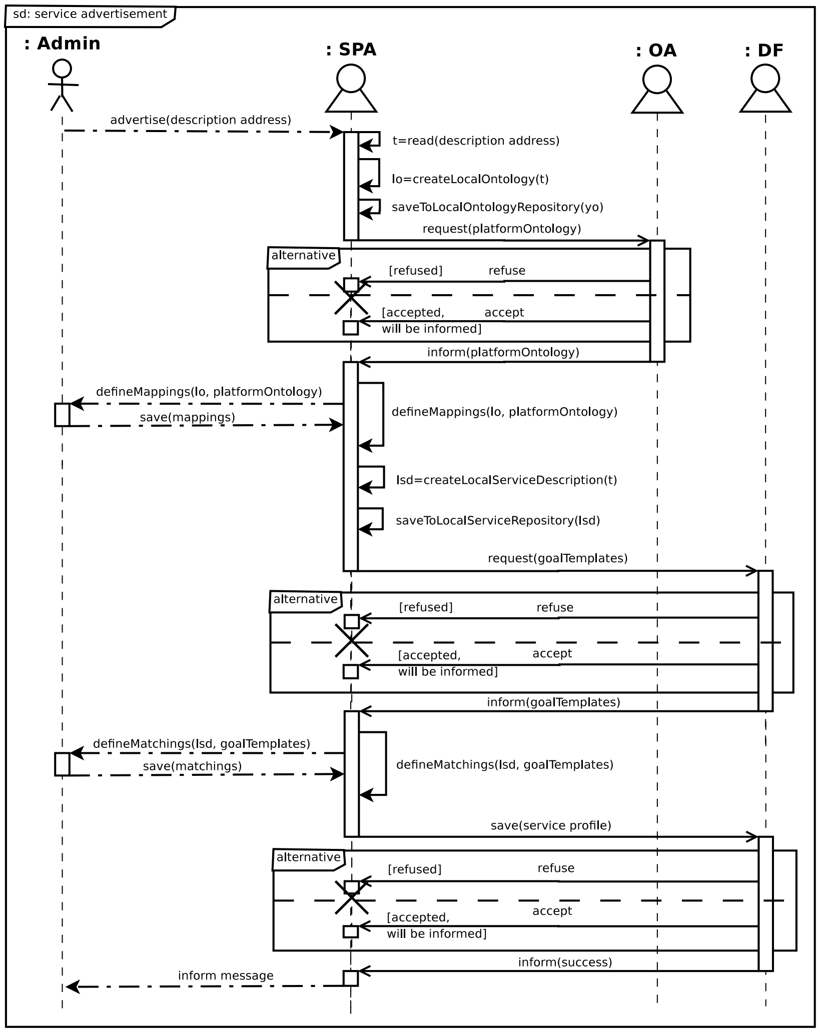
sd: service advertisement
: Admin
: SPA
: OA
: DF
advertise(description address)
t=read(description address)
lo=createLocalOntology(t)
saveToLocalOntologyRepository(yo)
request(platformOntology)
alternative
[refused]
refuse
[accepted, will be informed]
accept
inform(platformOntology)
defineMappings(lo, platformOntology)
save(mappings)
defineMappings(lo, platformOntology)
lsd=createLocalServiceDescription(t)
saveToLocalServiceRepository(lsd)
request(goalTemplates)
alternative
[refused]
refuse
[accepted, will be informed]
accept
inform(goalTemplates)
defineMatchings(lsd, goalTemplates)
save(matchings)
defineMatchings(lsd, goalTemplates)
save(service profile)
alternative
[refused]
refuse
[accepted, will be informed]
accept
inform(success)
inform message

Fig. 2. An External Service Advertisement Scenario

In case of a semantic service advertisement, the process depends on the interface language of the service: it might be OWL-S or another semantic service description language. Since OWL-S is the service description language of the platform it is not

[9] Agent UML, http://www.auml.org/, last access on July 24, 2007.

required to convert the description document when it is in OWL-S, only semantic compatibility is needed (mappings). But if it is in another language then the mechanism is similar to pure service advertisement: first the description document is converted to OWL-S, then semantic compatibility is obtained.

3.2 Service Discovery

In order to realize semantic service discovery, the platform services should be registered to a service registry which has the capability of matchmaking. Service requesters should query on this registry and have ability to interpret resultant service advertisements. As mentioned before, the DF of our platform stores capability advertisements of registered services as OWL-S profiles in its service repository and allows the semantic discovery of active services via querying these service descriptions.

The whole service execution process of the requester agent is handled by a plan which is based on a predefined generic semantic service execution plan for SWSA. This generic plan is discussed in our previous work [8]. When an agent (SRA) needs a service (a service to perform its goal), it loads that plan, and within the discovery phase it first forms a query. This query contains the capability description of the intended service and the degree of match to retrieve suitable service descriptions. This capability description is expressed in OWL-S profile and is valid if it conforms to the capability descriptions (goal templates) of the platform. The suitable communication protocol and content language for the client have already been designed and implemented for OWL-S services [5]. Then the DF receives the query request and using a semantic service matcher and a service repository it finds and returns the semantically appropriate services. Finally SRA receives the resultant services, selects one or more of them and then starts the engagement process with the providers of these selected services as the next step of its semantic service execution plan.

3.3 Service Engagement

After completion of the service selection, the requester-provider engagement process, which involves the negotiation on QoS metrics and agreement settlement. In our platform this phase is implemented in a simple manner. We just utilize some QoS parameters (like service cost, run-time, location, etc.) defined in various studies [2,15].

3.4 Service Enactment

After the engagement on the metrics of a semantic service between SPA and SRA, SRA initiates the enactment phase. SRA requests SPA to execute the engaged service using its service profile and the proper parameters. Then SPA finds the Service Matching Table using Matching tool and loads the plan within this table to enact the service. SPA first converts the parameters to the local ontology then converts them to the WSDL parameters using the mapping knowledge and the matching knowledge. Finally SPA prepares a SOAP message using these parameters and invokes the service. Result of the service execution is converted from its SOAP form into the platform ontology and sent to SRA. AUML sequence diagram of this scenario is shown in Figure 3.

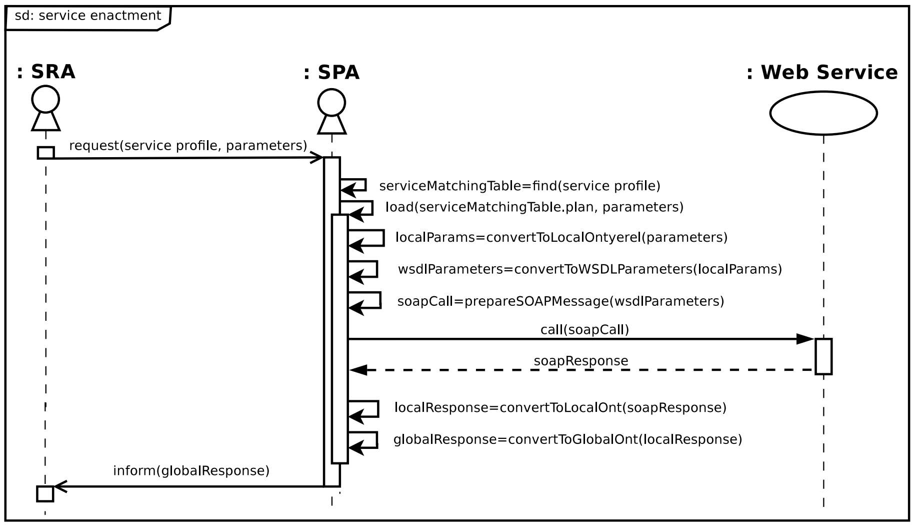
sd: service enactment
: SRA
: SPA
: Web Service
request(service profile, parameters)
serviceMatchingTable=find(service profile)
load(serviceMatchingTable.plan, parameters)
localParams=convertToLocalOntyerel(parameters)
wsdlParameters=convertToWSDLParameters(localParams)
soapCall=prepareSOAPMessage(wsdlParameters)
call(soapCall)
soapResponse
localResponse=convertToLocalOnt(soapResponse)
globalResponse=convertToGlobalOnt(localResponse)
inform(globalResponse)

Fig. 3. An External Service Enactment Scenario

4 Evaluation

The platform introduced in this study was employed during a commercial project in which design and implementation of a tourism system based on the SEAGENT [4] framework were realized. The project included adaptation of an existing hotel reservation system into the Semantic Web environment. The existing system was one of the products of a countrywide known software company which sells hotel automation systems[10]. The system had been previously based on the web service architecture and project aimed at providing semantic interfaces of the web services in use and realize an online system in which software agents reserve hotel rooms on behalf of their human users.

The development team used the architecture and semantic web service integration process discussed in this study during design and implementation phases of the project. Following is the feedback gained from the development team about utilization of our platform.

The developers in general found the architecture helpful especially in determining architectural roles and related agents in addition to the domain based role and agent specifications. They agreed that the architecture provided pre-defined constructs for their system and those constructs were used during the design of the system. Service Registry, Service Provider and Ontology Agents were all developed by the team using the SEAGENT framework. In the system, agents for the hotel clients were designed as service requester agents and they played the role of SRA.

The integration of the existing web services into the MAS was crucial and the team used our proposed service registration, discovery and execution dynamics during the re-

[10] Odeon Hotel Management Systems, http://www.myodeon.com/, last access on May 18, 2007.

alization of this integration. The team expressed that they previously had expected this task as a big challenge however they agreed that our service deployment mechanism simplified the work considering analysis, documentation and implementation issues. However, they found implementation of matching and mapping operations for the service integration in agent plans as a bit tricky and error prone. They also expressed that the policies used in service engagement were still too abstract and hard to implement.

5 Related Work

There have been a few partial implementations to integrate web services and FIPA compliant agent platforms. WSDL2Jade [14] can generate agent ontologies and agent codes from a WSDL input file to create a wrapper agent that can use external web services. WSDL2Agent [13] describes an agent based method for migrating web services to the semantic web service environment by deriving the skeletons of the elements of WSMO from a WSDL input file with human interaction. WSIG (Web Services Integration Gateway) [7] supports bi-directional integration of web services and Jade agents. WS2JADE [10] allows deployment of web services as Jade agents' services at run time to make web services visible to FIPA-compliant agents through proxy agents. But these tools only deal with the integration of agents and external web services and do not make use of semantic web technologies.

Some studies on integrating agent technologies with semantic web services also exist. For example the studies in [6] and [16] describe agent environments which use OWL-S (formerly DAML-S) to advertise descriptions of agent services in DF and to transport them with ACL messages. Dickinson and Wooldridge illustrate one approach using reactive planning to control web service invocation by BDI agents [3]. In these studies, the support for semantic web resources is limited and they do not provide complete system architectures. The most relevant work to our study is [12] wherein an agent framework is introduced for automated goal resolution on the semantic web. It uses WSMO-based technologies and tools. However it is not compatible with SWSA framework and the proposed architecture is hard to extend because of strict coupling to WSMO.

6 Conclusion and Future Work

An MAS platform for semantic service integration based on SWSA is discussed in this paper. We define a software architecture in order to provide concrete realization of the SWSA. Software agents are employed in automatic discovery and execution of the Semantic Web Services within this architecture. We also elaborate implementation of SWSA's sub-processes taking into consideration of the main components and their interactions of the defined architecture.

The protocols of the service engagement phase are currently in their preliminary state. Our first aim is to detail usage of the QoS parameters within the scope of the engagement protocols and fully employ service engagement during the agent - service interaction. We also plan to redesign and implement SPA component of the architecture so it would be a software tool for service providers by supplying a GUI for service integration. This would simplify semi-automatic service deployment for service owners. Therefore ontology mapping and process matching would be realized in a more comfortable and error-free way.

7 Acknowledgements

This work is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) Electrical, Electronics and Informatics Research Group (EEEAG) under grant 106E008.

References

1. M. Burstein, C. Bussler, M. Zaremba, T. Finin, M.N. Huhns, M. Paolucci, A.P. Sheth, and S. Williams. A semantic web services architecture. *IEEE Internet Computing*, Volume 9 Issue 5:72 – 81, 2005.
2. J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281–308, 2004.
3. I. Dickinson and M. Wooldridge. Agents are not (just) web services: investigating bdi agents and web services. In *SOCABE'2005 held at AAMAS'05*, 2005.
4. O. Dikenelli, R. C. Erdur, Ö. Gümüs, E. E. Ekinci, Ö. Gürcan, G. Kardas, I. Seylan, and A. M. Tiryaki. Seagent: A platform for developing semantic web based multi agent systems. In *AAMAS*, pages 1271–1272. ACM, 2005.
5. O. Dikenelli, Ö Gümüs, A. M. Tiryaki, and G. Kardas. Engineering a multi agent platform with dynamic semantic service discovery and invocation capability. In *MATES*, volume 3550 of *Lecture Notes in Computer Science*, pages 141–152. Springer, 2005.
6. Nicholas Gibbins, Stephen Harris, and Nigel Shadbolt. Agent-based semantic web services. In *WWW'03*, pages 710–717, New York, NY, USA, 2003. ACM Press.
7. D. Greenwood and M. Calisti. Engineering web service - agent integration. In *SMC (2)*, pages 1918–1925. IEEE, 2004.
8. Ö. Gürcan, G. Kardas, Ö. Gümüs, E. E. Ekinci, and O. Dikenelli. An MAS Infrastructure for Implementing SWSA based Semantic Services. In *Service-Oriented Computing: Agents, Semantics, and Engineering*, volume 4504 of *LNCS*, pages 118 – 131. Springer-Verlag, 2007.
9. M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with owls-mx. In *AAMAS'06*, pages 915–922, New York, NY, USA, 2006. ACM Press.
10. T. X. Nguyen and R. Kowalczyk. Ws2jade: Integrating web service with jade agents. In *SOCABE'2005 held at AAMAS'05*, 2005.
11. E. Sirin, B. Parsia, and J. Hendler. Filtering and selecting semantic web services with interactive composition techniques. *IEEE Intelligent Systems*, 19(4):42–49, 2004.
12. M. Stollberg, D. Roman, I. Toma, U. Keller, R. Herzog, P. Zugmann, and D. Fensel. Semantic web fred - automated goal resolution on the semantic web. In *38th Annual Hawaii International Conference*, 2005.
13. L. Z. Varga, K. Hajnal, and Z. Werner. *An Agent Based Approach for Migrating Web Services to Semantic Web Services*, volume 3192, pages 381 – 390. LNCS.
14. L. Z. Varga, K. Hajnal, and Z. Werner. *Engineering Web Service Invocations from Agent Systems*, volume 2691, pages 626 – 635. Lecture Notes in Computer Science, January 2003.
15. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *WWW'03*, pages 411–421, New York, NY, USA, 2003. ACM Press.
16. Y. Zou. *Agent-Based Services for the Semantic Web*. PhD thesis, the Faculty of the Graduate School of the University of Maryland, 2004.