Inter-organizational Interoperability through integration of Multiagent, Web Service, and Semantic Web Technologies[*]

Paul Karaenke[1], Michael Schuele[1], András Micsik[2], Alexander Kipp[3]

[1]University of Hohenheim, Information Systems 2, Schwerzstr. 35, 70599 Stuttgart, Germany, {karaenke, mschuele}@uni-hohenheim.de
[2]MTA SZTAKI, Budapest Lágymányosi u. 11. H-1111, Hungary, micsik@sztaki.hu
[3]High Performance Computing Center Stuttgart, Nobelstr. 19, 70569 Stuttgart, Germany, kipp@hlrs.de

Abstract. This paper presents a software architecture for inter-organizational multiagent systems. The architecture integrates Web service technology into multiagent systems to overcome the technical interoperability problem of current multiagent systems in the fast growing service-oriented environments. We integrate Semantic Web technology to make multiagent systems semantically interoperable. We address the problem of interoperability regarding interfaces, messaging protocols, data exchanged, and security whilst considering a dynamic e-business environment. The proposed architecture enables service virtualization, secure service access across organizational boundaries, service-to-agent communication, and OWL reasoning within agents.

Keywords: multiagent systems, web services, semantic web

1 Introduction

In recent years, service-oriented computing (SOC) has lead to significant transformations of industrial software architectures. SOC is in particular aiming at more flexible and open architectures for cross-organizational applications. It has originated a remarkable technology stack for Web services (WS) and respective standards which all contribute to the interoperability of systems. The same cannot be said for multiagent systems. However, it has been argued that both technologies "need each other" [5]. Both are concerned with problem solving by distributed systems, but focus on different approaches and offer divergent capabilities: WS-* standards facilitate the building of secure, robust and reliable virtual organizations (VOs) to solve problems with distributed resources, but lack the capability to react or adapt to undesired conditions and changing requirements in dynamic environments.

Multiagent technology, in contrast, offers the capability for flexible and adaptive problem solving behavior both on single agent and multiagent level, but lacks

[*] Appeared in: Springer Lecture Notes in Business Information Processing Volume 98, 2012, pp 55-75

reliability, security, and robustness [5]. Thus, combining WS and multiagent technologies could make use of the advantages of both technologies while avoiding their respective drawbacks.

WS specifications foster interoperability on the *technical* level regarding interfaces (e.g., WSDL) and messaging protocols (e.g., SOAP). However, specifications of the data exchanged (i.e., message content) is beyond the scope of WS specifications. The Semantic Web (SW) approach, in contrast, focuses on *semantic* interoperability. Thus, we propose an approach combining multiagent, WS, and SW technologies and a respective software architecture for inter-organizational multiagent systems. We address the problem of interoperability regarding interfaces, messaging protocol, data exchanged, and security. The contribution of this research is a software architecture for inter-organizational multiagent systems. This architecture enables service virtualization, secure service access across organizational boundaries, service-to-agent communication, and OWL reasoning within agents.

The remainder of this paper is as follows: section 2 defines the architectural requirements and discusses related work. In section 3, we describe our approach for integrating multiagent and Web service technologies. Section 4 describes the integration of multiagent and Semantic Web technologies. We provide an evaluation of the architecture in section 5. Section 6 summarizes the result.

2 Requirements and Related Work

2.1 Requirements Analysis

For the targeted interoperability of agent-based systems, agents must have adapters to WS systems; i.e., agents need capabilities for interactions with systems following the service-oriented architecture (SOA) paradigm. In addition, the inter-agent communication has to be conceptually based on standardized agent communication mechanisms (e.g., [3]). Communication mechanisms of agents have to be integrated with Web service technologies to (i) ensure the preservation of the SOA properties interoperability, reliability, security, and robustness for the integration and to (ii) enable the utilization of existing SOA infrastructures.

Communicating components need to have a common understanding of exchanged messages. This means at least, that exchanged messages can be enriched by annotations that refer to shared ontologies. Semantic technologies support the agents having behaviors with reasoning capabilities about the current agents' environment, the internal status, and especially messages received from other agents. Table 1 summarizes these requirements in a qualitative form following the IEEE software requirements specification [10] according to the recommended basic issues (functionality, external interfaces, performance, attributes, design constraints imposed on an implementation).

Table 1. Requirements.

Issue	Description
Functionality #1	Provision of inter-organizational communication interfaces and messaging protocols.
Functionality #2	Provide agent capabilities for calling Web services.
Functionality #3	Provide agent capabilities for providing Web service interfaces.
Functionality #4	The agents' internal reasoning is linked to semantically annotated data utilized in inter-organizational communication.
External Interface #1	The data exchanged between organizations is semantically annotated.
External Interface #2	The agent-to-agent communication via inter-organizational communication interfaces is based on Web service technology.
Performance #1	Throughput should not essentially deviate from intra-organizational communication.
Performance #2	Availability should not essentially deviate from intra-organizational communication.
Performance #3	Response time should not essentially deviate from intra-organizational communication.
Attributes #1	Ensure secure inter-organizational communication.
Design Constraints #1	Inter-organizational communication interfaces are based on Web service standards.
Design Constraints #2	Inter-agent communication has to be conceptually based on standardized agent communication mechanisms.
Design Constraints #3	Semantically annotated data bases on Semantic Web standards.

2.2 Related Work

Related research exists in the areas of translating agent to WS messages and vice versa as well as in the area of integrating Semantic Web languages in agent communication and reasoning.

The coupling of agents and WS resources in a similar approach is investigated in [21], though the authors remain on a very high level of abstraction and do not consider agent-to-agent communication based on WS technology. The Web Service Integration Gateway (WSIG) [6][12] is an official Jade plug-in, which provides bidirectional invocation facility, by which Jade agents can call WSs and WS clients can call Jade agent services. The connection of agents and WSs is implemented using elaborate on-the-fly translation between agent messages and SOAP messages. However, WSIG cannot connect agent platforms via SOAP and does not allow transparent wrapping of agent message content into SOAP, thus it fails to support the mentioned scenarios (dynamically changing business partners, secure communication, etc.).

The AgentWeb Gateway [24] is a middleware between agent platforms and WS platforms, which provides various transformation mechanisms between the two worlds: agent platforms and web services. Therefore, it enables integration of the two worlds without changing existing specifications. As part of the solution, the gateway provides SOAP to ACL and ACL to SOAP protocol converters. One of our goals is to achieve SOAP-based communication among agent platforms (External Interface #2). However, dynamic aspects as required within typical inter-organizational scenarios are not considered, such as the dynamic and transparent substitution of collaboration partners as well as the transparent integration of security related issues. These exemplary issues play a fundamental role in inter-organizational environments, since in particular the substitution of service endpoints typically enforces a significant effort to build up an according, trusted environment. Furthermore, implementation details or source code are not available for this work.

Paurobally and Jennings [23] use WS protocols to describe speech acts and implement the contract net protocol for WSs. This approach requires a complex re-engineering of existing WSs to handle speech acts and agent coordination protocols, whereas multiagent platforms already include this functionality and provide a variety of well established coordination protocols. Furthermore, software agent properties like pro-activeness and goal-orientation are not considered in this approach.

[7] gives an overview of different approaches to directly integrating FIPA communication and Web Service Technologies, and collects requirements for such an integration. In prior work, we have evaluated the mentioned approaches [18] for this aspect of our architecture and found that most of them are not available for re-use (reasons are given below).

The approach of Soto [23] falls closest to ours, but in this case the contents of FIPA envelopes are mapped to WS-Addressing headers, thus it is unavailable for use by the transport layer itself. Later it will be explained that WS-Addressing headers are used in a different way in our environment.

WS2Jade [21] is an integration approach focusing on closely coupling Jade agents and Web Services by representing a web service by a gateway agent. A new gateway agent is created automatically for a WS by WS2Jade (which could lead to performance and management issues in our scenario). As WS2Jade focuses on agents executing web services, it is not suited for bi-directional agent-to-agent communication via SOAP either.

Moreira et al. propose the AgentSpeak-DL dialect, which extends AgentSpeak towards Description Logic (DL) [19]. However, AgentSpeak-DL is a specific language and its underlying DL is less expressive than OWL Lite and OWL DL. Therefore, it does not allow for using existing OWL ontologies, but required modifications to these. AgentOWL is a Jade plug-in, which enables agents to exchange SPARQL queries and RDF triples in ACL messages [17]. It also contains an embedded inference engine and connection possibility with remote knowledge bases. However, AgentOWL is not integrated with a BDI implementation, and furthermore its interoperability with external ontologies is unsuitable for our purposes. Nuin [2] is an agent framework designed for practical development of agents in SW applications based on BDI principles. The agents running in Nuin may have access to different knowledge sources, including RDF and OWL ones. However, Nuin does not support RDF knowledge sources fully. In contrast to Nuin, our approach establishes a direct connection between the agent belief base and the underlying OWL knowledge base, which supports both accessing and modifying OWL facts.

JACK Intelligent Agents [9] is a framework for multi-agent system development, which is built using the BDI principles, so agents can manage beliefs, desires and can also plan their actions to implement their desires. It also provides its own Java-based plan language and graphical planning tools. Although JACK has an extension that enables it to communicate with other FIPA agents, the framework is a commercial, closed software, and thus could not be used in our case.

3 Integrating Multiagent and Web Service Technologies

In this section, we describe our approach to integrate Web service technologies into multiagent systems. Firstly, we describe the overall approach of coupling multiagent and WS technologies. Secondly, we give details about secure inter-organizational communication in our approach. Finally, we give details about secure agent communication in inter-organizational settings utilizing the two former approaches.

3.1 Encapsulation of Web Service Resources

We propose an approach for coupling of WS and agent technology in a head body architecture [26][8]. The head body paradigm implies a conceptual separation of a software agent into two parts – head and body. The agent's head is used for interactions with other agents. This includes reasoning about interactions such as participating in cooperative processes for problem solving. The body is encapsulating any other (domain) functionality of an agent [26][8]. The head body paradigm is used in the approach shown in Fig. 1. WS resources that are represented by agents are part of the body. The agent's core capabilities are implemented in the head; i.e., interactions and especially coordination with other agents in the agent society. The agent communicates with the encapsulated WS via WS sensor and WS effector.

On the conceptual level, agent-to-agent communication is based on FIPA communication standards (e.g., [3]). On the technical level, agent-to-agent communication is based on WS technologies and standards. Therefore, the agents can be used in existing WS infrastructures and systems. The presented approach of coupling WS and agents allows the utilization of multiagent coordination protocols for the coordination of existing WSs in existing infrastructures. A WS which is represented by an agent can transparently be invoked by WS clients. The agent can evaluate the invocation requests and can reason whether an invocation of the encapsulated WS is in accordance with its own goals. If the invocation request is opposed to the goals, the agent can intercept the invocation and the encapsulated WS is not invoked. Further, the agents can pro-actively work towards the goals; e.g., maximizing revenue for encapsulated resources by establishing Service Level Agreements (SLAs).

A special case of the invocation of a Web service encapsulated in an agent's body is the processing of invocation requests from external callers. In this case, the agent acts as a proxy for the Web service to the external environment. This case is especially important for the integration of WS and multiagent technology, as the encapsulation is transparent to the external caller; i.e., the approach can be used in existing WS infrastructures and for existing Web services.

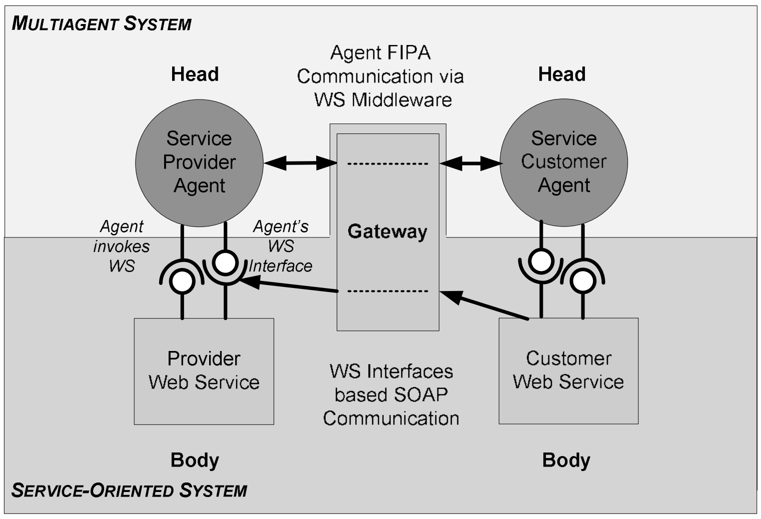

Fig. 1. Head Body Architecture

An external invocation request of an encapsulated Web service is initiated by either another agent or an external Web service client. The agent has full control over the encapsulated Web service and can decide, based on semantic reasoning, if an invocation of the Web service is compatible with the agent's own goals. That is, the agent can decide if it invokes the Web service or not. The result of the invocation is then delivered to the component that has requested the invocation.

Practical examples for reasoning about service invocations fall into the areas of task prioritization and business partnership. The type of task requested is naturally extractable from the service request. Furthermore, our framework injects the business context into the service requests, namely the identifier of the customer and the contractual context of the request, as further discussed in 3.2. By this information, the agent can decide to hold or completely re-order the execution of incoming service requests, and here the semantic description of tasks, business partners and priorities are utilized in conjunction with reasoning inside agents.

3.2 Secure Inter-organizational Communication

In order to allow for a seamless integration of components, a corresponding flexible and adaptive messaging infrastructure, commonly titled as "Enterprise Message Bus", has to be provided. We address this requirement by a set of components referred to as the Gateway Toolkit which provides such a messaging infrastructure by allowing for a "double-blind" virtualization approach [16]. On customer side, the Gateway Toolkit allows to hide the concrete service provider (SP) which allows describing corresponding workflows in a more abstract manner. Additionally, the customer can easily change service providers by adapting the routing information of the Gateway Toolkit infrastructure whilst not affecting the corresponding workflows.

The SP can easily hide the underlying service infrastructure by providing virtual, callable service endpoints to potential customers. The major benefit of the service provider hereby is that he is now able to adapt the underlying service infrastructure whilst not affecting the corresponding service customers. The SP is additionally enabled to involve third party SPs for particular sub-tasks without affecting the customer. Such a realization provides benefits for both sides, the service providers as well as for the service customers. Service provider can easily provide their "products", e.g. in the form of combined services, in such a way that potential service customers can integrate these services in their own products. This is done in an abstract manner which means in particular that no implementation details of the underlying service implementation needs to be considered.

Through using "double-blind" virtualization mechanisms, i.e., by deploying the gateway on both consumer *and* SP side, it is possible to alter resources *and* providers without affecting the calling applications. Fig. 2 presents a sample deployment of the Gateway infrastructure. The dashed line denotes an indirect link (as the gateway extends the message channel).

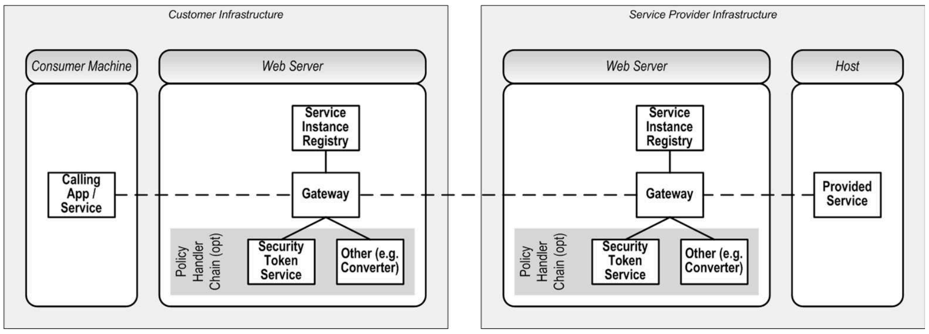

Fig. 2. Sample deployment of the Gateway infrastructure

This deployment of the gateway allows a transparent and secure interaction channel for the involved components. In particular, the gateway allows the provision of virtual endpoints via which the corresponding components are able to interact with each other in a secure way without the need to explicitly consider the corresponding security as well as the according WS standards. The interaction is done completely transparent for the components whilst considering dynamic e-business needs; e.g., the possibility to change service provider during runtime, transparent usage of resources whilst considering accounting and secure and reliable communication.

The Gateway mediates the communication between the front end WSs of the two domains. Each front end authenticates itself to their respective Gateway. The Gateway allows the invocation of virtual service endpoints by resolving these virtual to concrete endpoints via the service instance registry (SIR). The SIR also provides additional metadata such as the gateway endpoint that the message has to travel through, as well as the endpoint of the security token service (STS) where tokens affiliated with this service can be requested.

Virtual addresses used in SOAP messages can be translated dynamically to appropriate real services. This enables service fault management, on-the-fly reconfiguration, and other advanced solutions to enhance reliability of the service environment. For example, a Web service can be moved to a different host very easily, only its virtual address has to be remapped to the new real endpoint address inside the Gateway.

The STS issues claim-based tokens to authenticated users, respectively software components, and is also involved in the process of establishing federation with the STS in the SP domain. The consumer-side role of the STS issues tokens that are necessary to pass the security check on the service side. The tokens are generated based on the information that is extracted from the service call message. The service-side role of the security token service acts not as token issuer but as verification instance for security tokens that are attached to the incoming message. It hence has the role of a policy decision point (PDP). In the example, the consumer requests a service from his own SIR by providing an URN (uniform resource name) and the SIR returns the virtual address along with the endpoints of the gateway and the STS. The provider side SIR will convert the server side virtual endpoint to an actual endpoint where the client request can be satisfied.

This dynamic and flexible infrastructure enables service consumers as well as service provider to react easily on dynamic changes within a service environment, whilst taking account security issues in a transparent way as well. In particular the latter is a significant issue when referring to common ways in establishing trusted service environments, namely Virtual Private Networks (VPN). These networks are characterized by a static end-to-end connection, which makes it difficult to dynamically change the according service endpoints. In particular these networks do not allow for the transformation and adaptation of the according messages, so in case of a service provider replacement both, the according service endpoint as well as interface descriptions have to be considered by the invoking application, whilst the introduced gateway infrastructure allows for the invocation of virtual service endpoints by allowing for the transparent adaptation of the according messages within the gateway environment.

3.3 SOAP-based Agent Communication

For multiagent systems, FIPA provides specifications in the area of agent communication and agent message transport. The Agent Message Transport group defines ACL (Agent Communication Language), envelope representations, and the transport protocols that can be used to transfer agent messages between hosts. Currently IIOP and HTTP protocols are supported for agent message transport (WAP support is experimental). The problem concerning multiagent and WS integrations is that the use of IIOP and WAP is declining, while the HTTP based MTP (Message Transport Protocol) is incompatible with SOAP (the most commonly used WS messaging protocol) as it uses the MIME multipart construct to transfer the agent message envelope and agent message body parts, while SOAP has its own XML-based way to embed arguments into the request.

Furthermore, the communication between multiagent groups, containers or platforms also raises issues when this communication crosses organizational boundaries. Organizations often aim at providing a unified messaging architecture, which can be administered and monitored easily and centrally. The aim of such messaging architectures is to ensure the reliability, flexibility, and security of message transfers. Since we propose a mixed environment of WSs and agents, a natural solution is to transfer messages using SOAP and WS-* standards. Therefore, we utilize SOAP as message transport between multiagent platforms.

Uniform transportation of agent and WS messages simplifies system administration and enables common mechanisms to be introduced in routing and delivery. This is achieved by adding support for a new MTP to agent platforms. The SOAP MTP add-on [18] is a pluggable driver for sending and receiving SOAP messages and translating them to/from internal agent message format. Each agent platform uses the SOAP MTP add-on configured with a virtual endpoint address, which is mapped to the agent platform address in the Gateway component. The virtual endpoint address is also advertised in registries and directories outside the organizational domain, so that external entities will use the virtual address to reach the agent platform.

Agent platforms can be operated in separate organizational domains. Inside each platform the communication between agents is usually not supervised and not restricted. Similarly, agents can access WSs freely inside the domain. However, the communication between agent platforms has to be supervised, according to current policies of the embedding domains.

Other approaches like the AgentWeb Gateway [24] or WSIG [6] also provide basic support to enable agents to communicate via WS technologies. However, these approaches lack in facing these essential e-business requirements thus enforcing the agent developer to adapt the setup of the corresponding framework every time an evolution step has been processed (cf. section 2). To this end the interaction via virtual endpoints allows the adaption of the communication infrastructure during runtime; e.g., in the case an agent has to be replaced by another without affecting the remaining involved agents at any time, which is an essential need for dynamic e-business environments.

Messages between administrative domains are sent and received by the Gateway of each domain (Fig. 3). In our example the Jade agent platform [11] is used. The following steps are executed when sending a message to a remote agent platform: (1) the consumer agent addresses the message using the virtual endpoint address of the remote agent on the SP side. (2) The Messaging Service detects that this address belongs to the SOAP MTP, and forwards the message to the SOAP MTP add-on for delivery. (3) The SOAP MTP client prepares the SOAP message, and delivers it to the virtual address of the remote agent, but the outgoing message is actually caught by the local Gateway. (4) The local Gateway identifies the recipient SP using the SIR, and arranges for a security token with the STS of both sides. (5) The message is sent to the Gateway at SP side. (6) The SP Gateway checks the access rights for the service, decrypts the message, then finds the real endpoint service using the SIR, and calls the endpoint of the Jade platform. (7) The SOAP MTP of the SP's Jade platform reconstructs the original agent message and passes it to the internal Messaging Service, which finally delivers it to the recipient agent.

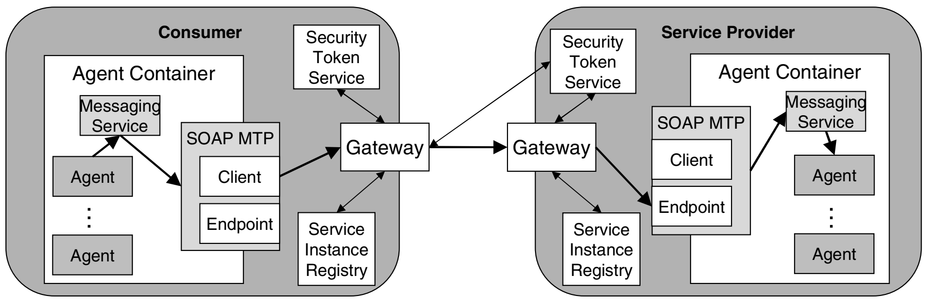

Fig. 3. Inter-organizational Agent Communication

4 Integrating Multiagent and Semantic Web Technologies

This section describes our approach to integrate Semantic Web technologies into multiagent systems. We give a conceptual overview and provide details about the interactions of different system components. Further, we provide details about the implementation.

4.1 Concept and Architecture

The belief-desire-intention (BDI) approach [1] is the most common architecture for deliberative agents, agents who deliberate over symbolic knowledge to reach given goals [28]. That is, the BDI architecture facilitates goal-driven system behavior. The model consists of the following concepts: beliefs capture informational attitudes realized as a data structure containing current facts about the world. Desires capture the motivational attitudes that form concrete goals, if an agent has potentially the chance to fulfill the desire. Intentions capture the deliberative attitudes realized by reasoning to select appropriate actions to achieve given goals or to react to particular situations.

A BDI agent is equipped with sensors to assist it on its environmental awareness, and effectors to impact the environment by actions. A reasoning mechanism between the sensors' input and the effectors' output deduces the necessary actions for achieving the agent's goals. The agent acquires new beliefs in response to changes in the environment and through the actions that it performs as it carries out its intentions [1]. Thus, the BDI agents allow reasoning regarding decisions to determine which, possibly conflicting, business goals can be achieved and how the agent is going to achieve these goals. For example, for an agent representing a resource of our case, beliefs correspond to the state, capabilities, and SLAs of the resource; desires represent the business goals of the resource provider, while intentions result from a collection of possible decision mechanisms to select and execute requests to use the resource.

In addition, the BDI concept has been integrated with explicit semantics: the agent's beliefs, stored in the agent's beliefbase, are completely based on semantic data. Further, semantic reasoning is applied to derive new knowledge – especially required actions to reach goals – based on the semantic beliefs. Conceptual definitions of SLA parameters, metrics, and economic values as well as resource characteristics are given in an OWL DL ontology [27]. New data arriving to the agents are inserted into the knowledge base, which is automatically enriched using DL reasoning. Agents can then retrieve the results of reasoning via the beliefs. This provides essential support towards the targeted technical interoperability over organizational boundaries, representing real-world business relationships.

Fig. 4 shows the interactions of the Semantic BDI Agent's internal components for semantic BDI reasoning: based on an internal (step 1) or external (step 2) event, the agent first stores new facts into its beliefbase (step 3). The agent utilizes semantic reasoning to assess the event, deriving new knowledge (step 5-10) and especially appropriate intentions to achieve the agent's goals (step 11). These intentions lead to actions (step 12) which potentially include interactions with external components via the agent's effectors (step 13).

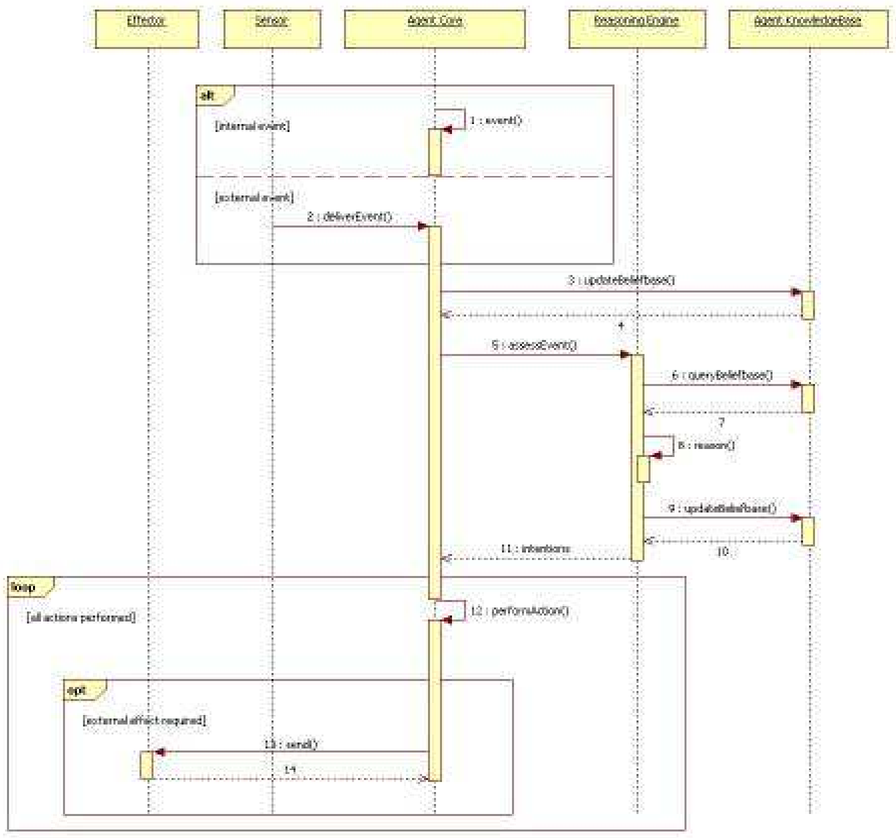

Fig. 4. Semantic BDI Reasoning Sequence Diagram

4.2 Realization

An embedded OWL engine provides the core semantic functions inside the agents. The OWL engine is connected to the BDI agent via the beliefbase. The implemented BDI agent plans add or modify facts in the semantic database, which activate any OWL DL reasoning, SWRL or other rules inside the semantic core. The BDI agent core (Jadex [13] in this case) polls dedicated beliefs, which are actually stored in the semantic database. The following cut-out of the agent definition file (ADF) of our Jadex agents shows the described mapping relation between facts in the knowledgebase and the agent's beliefs. The reference to the semantic knowledge base itself is integrated as a dedicated belief. The knowledge base receives a reference on the beliefbase to realize a bidirectional mapping between knowledge base and beliefbase.

```
...
<belief name="kb" class="JadexReasoner">
     <fact>
         new JadexReasoner(„http://../url/for/ontology", $beliefbase)
     </fact>
</belief>
...
<belief name="availability" class="Float" exported="true">
    <fact evaluationmode="dynamic">
         $beliefbase.kb.getPropFloatValue($agent.getName(), "#avail")
    </fact>
</belief>
...
```

Thus, when reasoning changes the semantic representation of the agent beliefs inside the semantic core, it can trigger a goal via the BDI beliefs. Finally, when goals activate selected plans, the semantic core is updated and the loop starts again. We experimented with prototype implementations for the embedded lightweight semantic core using the Jena SW toolkit, Jena built-in rules and Pellet OWL reasoner (applicable as SWRL rule engine as well). Both solutions provided a small and effective extension to our BDI agents.

In order to exchange semantic data, the components can apply two methods. RDF can be exchanged as plain text (the N3 notation is more convenient during development). Further, semantic annotations can be used with existing XML message formats. An example for the latter is the Semantically Annotated SLA (SA-SLA) format, used for the negotiation of Service Level Agreements (SLAs) between agents [20]. SA-SLA takes an existing XML representation of SLA and connects XML elements to corresponding ontology concepts. Therefore, the common interpretation of loosely defined XML elements is ensured. In contrast to existing approaches (e.g., [17]), our approach allows utilization of OWL not only as a content language for agent messages but also for data exchange with other software components (e.g., WSs). In addition, it enables to determine appropriate actions to reach the agents' goals based on semantic reasoning.

5 Evaluation

To demonstrate the applicability and usefulness of our approach, we perform two types of evaluation. Firstly, we analyze the performance of the secure inter-organizational agent communication to provide evidence that our approach does not result in an inappropriate overhead. Secondly, we provide a detailed use case description for our approach, which also includes details about the implementation. We conclude this section with a discussion of the benefits and drawbacks of our approach and its potential impact on inter-organizational interoperability.

5.1 Performance Evaluation

This section provides a performance evaluation of the inter-organizational agent communication. The goal of this evaluation is to provide evidence that the proposed approach does not cause an inappropriate computational overhead. The experiments evaluate the proposed SOAP MTP and Gateway infrastructure in comparison to conventional agent message transport via RMI and HTTP regarding performance in terms of communication time. In the following experiments we compare the agent message transport (1) locally via RMI, agent-to-agent communication on one platform, (2) distributed via HTTP, agent-to-agent communication on different platforms/machines via HTTP, (3) distributed via SOAP MTP, agent-to-agent communication on different platforms/machines via SOAP MTP, and (4) distributed using Gateway infrastructure, agent-to-agent communication on different platforms/machines via SOAP MTP using the Gateway infrastructure.

The setup of experiment1 contains two agents which interact accordingly with the FIPA Request Interaction Protocol [4]. Agent1 constitutes the initiator; it sends a *request* to Agent2, the participant. Agent2 replies to the request with an *inform* message to Agent1. The experiment repeats this process 1,000 times for each type of agent message transport. The duration of the protocol execution is measured for every iteration. Fig. 5 shows the results of experiment1 in one chart for each type of agent message transport. Table 2 shows the mean execution time for every type of agent message transport.

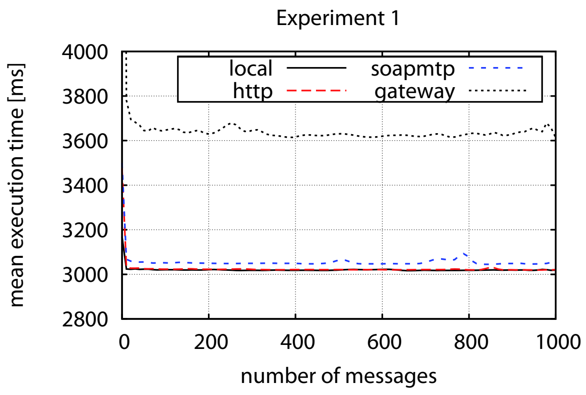

Fig. 5. Mean execution time in experiment1

Table 2. Mean execution time in experiment1

	Local	HTTP	SOAP MTP	Gateway
Mean execution time [ms]	3,019	3,023	3,053	3,641

In experiment2, number of agents is extended to 20. Ten agents act as initiators and ten as participants. Each initiator sends 100 requests to each receiver. The duration of the protocol execution is measured for 10,000 interaction iterations. Fig. 6 shows the mean execution time over all initiator agents in experiment2. Table 3 shows the mean of the protocol execution time of every initiator agent for every type of agent message transport and the mean over all initiator agents.

Table 3. Mean execution time in experiment2

	Local	HTTP	SOAP MTP	Gateway
Agent1 [ms]	4,376	4,458	4,433	4,806
Agent2 [ms]	4,365	4,511	4,409	4,842
Agent3 [ms]	4,304	4,476	4,412	4,828
Agent4 [ms]	4,373	4,435	4,447	4,825
Agent5 [ms]	4,368	4,478	4,376	4,914
Agent6 [ms]	4,366	4,488	4,420	4,956
Agent7 [ms]	4,325	4,472	4,434	4,922
Agent8 [ms]	4,368	4,477	4,414	4,946
Agent9 [ms]	4,375	4,511	4,420	4,951
Agent10 [ms]	4,338	4,472	4,409	4,916
mean Agent1 – Agent10 [ms]	4,356	4,478	4,417	4,891

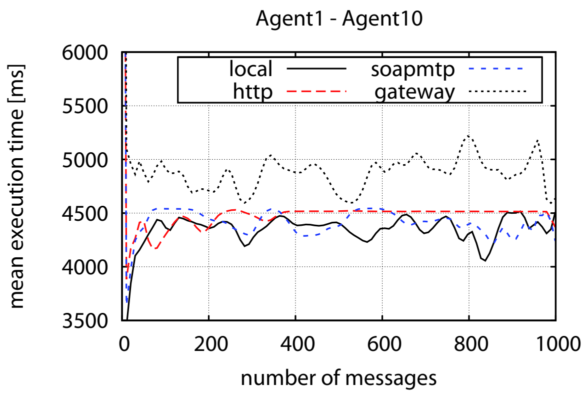

Fig. 6. Mean execution time over all agents in experiment2

5.2 Use Case

In this section, we provide a detailed use case description for a scenario from the airport logistics domain to show the applicability and utility of our approach. Further, we provide details about a prototype implementation.

Business Background. Airports are subject of an on-going transformation from monolithic, hierarchical organizations to networks of multiple companies that (1) either cooperate closely to offer services to its customers or (2) allocate resources by market-based mechanisms in competition [14]. For instance, ground handling services are no longer offered only by the airport exclusively but also by third parties, even including airlines, both located at the airport. In these cases, the airport provides only basic infrastructure and services such as buildings, flight plan information, access to power supply and telecommunication networks etc.

Due to ad hoc changes in the flight plan (e.g., due to delays), there can be temporary resource shortages for ground handling service providers. Thus, these providers need to outsource some tasks to another service provider that has sufficient resources for the time frame in question, i.e., the problem the provider needs to solve goes beyond a local optimization problem. The interoperability of the service providers' information systems, especially regarding availability information for outsourcing requests, is a key aspect of decision support for ground handling dispatchers. This interoperability is also required to enable resources of a ground handling service provider to receive task information from, respectively provide execution information to other service providers.

Current ICT systems at airports realize interoperability by increasing the coverage of the airport's enterprise resource planning (ERP) system as far as possible, thus replacing legacy systems and making interfaces obsolete. In network organizations, however, this approach is no longer feasible, since network participants will not make short-term, risky investments in proprietary, hence airport-customized ERP systems. In the airport management domain, de-facto standard software packages do not exist, which makes interoperability even more challenging.

Connectivity addresses the capability of ICT systems to allow an easy entry to the airport system as well as an easy dissolution of the connection. This capability supports new business models of service providers that participate flexible and temporarily in such virtual organizations. Both issues call for a machine-readable, unambiguous and inter-organizational representation of the common domain concepts, the actors, their goals and service capabilities.

Airport operations are subject of extensive security requirements (e.g., security checks, hand baggage restrictions, customs, anti-terrorism measures) as well as legislation (e.g., passenger rights) that either can (1) require to exchange a specific information with third parties or (2) prevent exchanging and accessing all flight and passenger-related information within the process.

The use case setup, which has been developed in the EU project BREIN (http://www.eu-brein.com), is as follows: Stuttgart Airport has about 400 flight movements per day. Most of the flights arrive and depart in two peak time periods: in the early morning and late afternoon. Fig. 7 shows an example of the flight movements over time. However, all planned activities can be subject of internal (e.g., resource failure) and external disturbances (e.g., delay of arriving flight) that affect, delay or constrain the process. These deviations cannot be foreseen; though their probability can be forecasted to some extend and thus considered in planning.

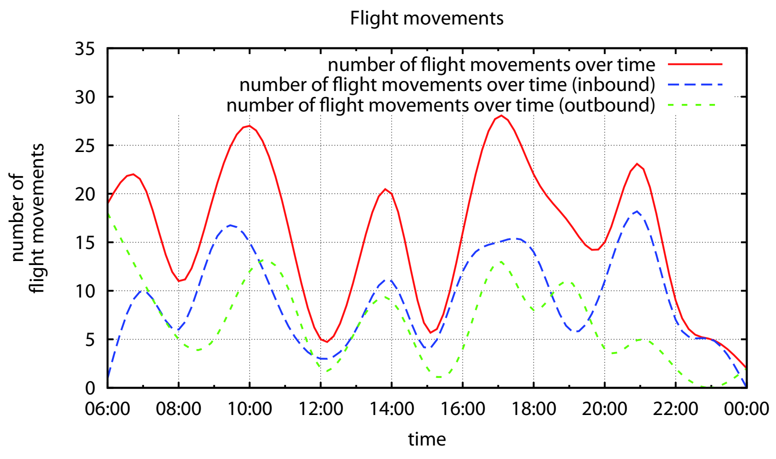

Fig. 7. Flight movements over time

Scenario Description. The service network consists of airlines as customers and several ground handling service providers. Actors on the provider side are ground handling companies and their resources (exemplarily limited to busses, baggage carts and baggage handling staff). The airlines have SLAs with the ground handling companies regarding ground handling services in order to dispatch the airlines' aircrafts. The resources provide atomic services for their ground handling company. In the case of a resource bottleneck, the ground handling companies are able to outsource tasks to another ground handling company. Fig. 8 shows the service network in a graph. Nodes represent actors, edges represent services, *x* denotes the tier of the service network.

The key data of the scenario is shown in Table 4. The three considered types of resources are able to provide five different types of services. The number of ground handling companies, airlines, and aircraft types is based on data from Stuttgart Airport for a single day.

The ground handling companies' dispatchers have pre-planned schedules of the tasks that have to be executed by the resources according to the incoming and outgoing aircrafts. The dispatchers can obtain dynamic flight plan information (e.g., flight delays) to reschedule resources accordingly. The resources inform the dispatchers about their current situation, task execution progress, delays, and malfunctions.

In case of a deviation of the pre-planned schedule, e.g., because of a delay of an aircraft, the dispatchers try to shift tasks to available resources to resolve any conflicts resulting from the deviation. If the own resources are insufficient, the dispatcher can outsource some of the tasks for the dispatching of the aircrafts to another service provider at the same airport which has sufficient resources for the time frame in question. Thus, the problem the provider needs to solve goes beyond a local optimization problem.

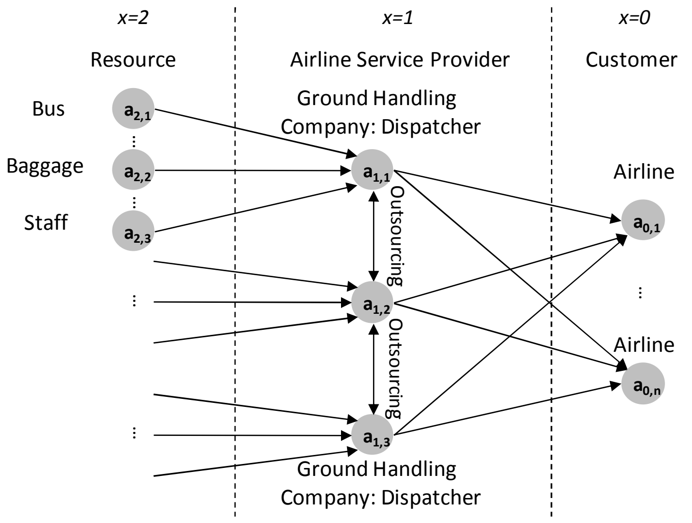

Fig. 8. Service network of customers, airline service providers, and resources

Table 4. Use case scenario key data

Parameter	Characteristic
Number of ground handling companies	3
Number of resource types	3 (bus, baggage, staff)
Number of resources	58
Number of service types (offered by resources)	5 (passenger transportation, deliver baggage to flight, deliver baggage to claim area, loading, unloading)
Planning horizon	full day
Number of airlines	15
Number of aircraft types	18

In this context our integration approaches for integrating Web service and Semantic Web technologies into multiagent systems are applied in order to fulfill the requirements of intra-organizational und inter-organizational task reallocation regarding interoperability, connectivity and security. The WS communication enables interoperability on the interface level while the Semantic Web technologies enable semantic interoperability regarding task, resource and SLA information based on shared ontologies. The Gateway Toolkit provides the required security infrastructure.

Realization. Each actor is represented as a Jadex software agent equipped with Jena SW toolkit as described in section 4.2. An agent platform with the respective agents and the Gateway Toolkit are deployed on three different machines for the different ground handling organizations. All communication between the organizations is routed through the Gateways. In addition, we have implemented a central visualization interface to which all agents report their status, movements, etc. A screenshot is shown in **Fig. 9**.

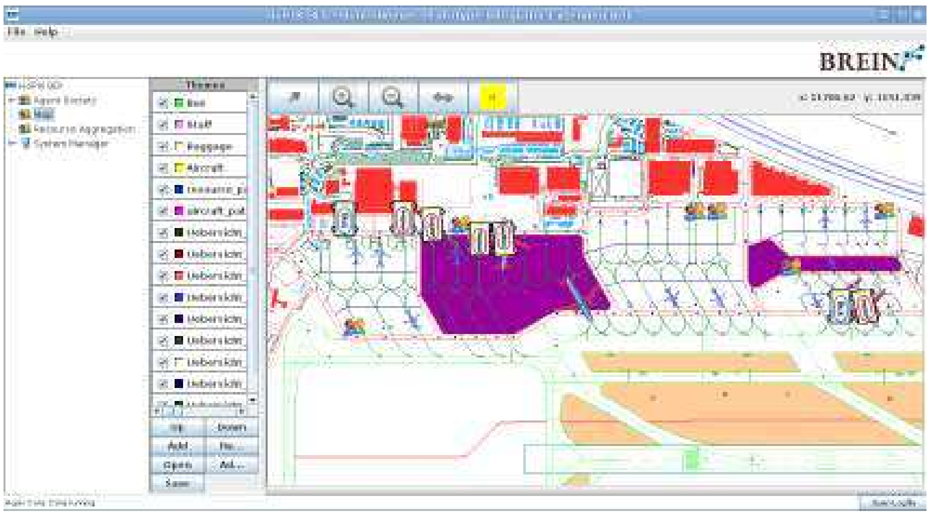

Fig. 9. Screenshot of the visualization interface

For intra-organizational and inter-organizational task reallocation, we apply a market-based coordination approach. The dispatcher agents can execute reverse auctions for tasks that are to be executed by its resource agents. These can answer the messages with bids on the atomic tasks. The dispatcher agent will select the appropriate allocation of bids and tasks by semantic reasoning and informs the resource agents accordingly. If the intra-organizational reallocation fails, the dispatcher agent tries to outsource the conflicting tasks to another ground handling company, executing reverse auctions with the other two dispatchers. These will disaggregate and forward the invitation to bid to their resources in a multi-tier interaction sequence, which we have described in previous work [15].

This behavior leads to different implications for the inter-organizational communication. The dispatchers have to be able to communicate with each other. This is done by pre-defined policies in the Gateway Toolkit. Once the execution of a task has been outsourced to another ground handling company, the client's dispatcher has to be able to communicate with the contractor's resource agent to coordinate and monitor the provision of the ground handling service. This can be realized by dynamically adapting the policies in the Gateway Toolkit.

Regarding the contents of the interchanged messages, our approach is based on semantic annotations. For example, the inter-organizational bids of the dispatchers are realized as SA-SLA templates; i.e., description of the services which the agents are willing to accept including information on agreement creation constraints. These SA-SLA templates contain references to ontology concepts as described in [20]. Thus, the receiving agent is able to use the semantic reasoning mechanisms to determine if and which of the received bids it accepts.

5.3 Discussion

The results of performance experiment1 show that the mean execution times of the local RMI transport, the distributed HTTP, and SOAP MTP transport differ in a small range of about 35ms. The gateway approach differs from the other approaches with a mean execution time difference of about 600ms (20%).

In the performance experiment with 20 agents, the mean execution time of the local RMI, distributed HTTP, and SOAP MTP transport approach increases to approximately 4,400ms. However, the mean of the gateway transport approach differs from the other approaches in experiment2 with only about 500ms (11%). An analysis of the charts of the single agents shows that the divergences from the mean are more explicit in this experiment. These divergences are caused by the mechanisms of the BDI framework and relativize the time differences regarding the type of agent message transport. This effect is also visible in the chart with the consolidated mean over all initiator agents. In the use case scenario, the additional overhead caused by the secure messaging infrastructure is acceptable with regard to the benefits of an inter-organization decision support system.

A key advantage of the presented architecture is a simple, yet powerful communication using a single message bus for both agents and WSs, gateways for the protection of organizational boundaries, and exchange of semantic content based on shared ontologies. An example for the successful application of all these benefits is SLA negotiation, where the SLA requests and offers can be exchanged using common semantics among several service providers. Furthermore, SLAs can be interpreted and reasoned about inside agents, enabling the use of agent cooperation mechanisms for SLA negotiation. Using the proposed architecture based on WS technology, agent messages can be transferred in a secured way, agent messages can be routed through gateways, and agent addressing can be virtualized; i.e., the agent platform can be dynamically relocated to a different address. The accessibility of agent platforms can be enhanced, as SOAP-based transport is more tolerant with firewalls and other security restrictions. Heterogeneous WS and agent environments may use a homogeneous message transport layer that reduces the complexity of system administration. It also enables secure inter-organizational transfer of agent messages between agent platforms, thus facilitating the advantages of both multiagent and WS technologies in a single environment. The utilization of explicit semantics further facilitates the semantic interoperability by incorporating domain knowledge in all phases of the service life cycle.

Furthermore, agents representing various resources of the airport may have their own, local reasoning support to enhance their operation. For example, at the lowest level, transport vehicles have maintenance regulations. This means that the vehicle has to visit the service station at regular intervals. Additionally, the vehicle or its driver may detect types of malfunction, which have typical repair time. The schedule of the next maintenance and the estimated time while the vehicle would be out of service affects the overall schedule of its service provider. The knowledge about previous maintenance, detected errors, and detected parts to replace combined with forthcoming jobs can be used to suggest time slots for the next maintenance of the vehicle, when it is less disturbing for the customers and yet keeps the vehicle in good condition. A simplified solution for this use case has been implemented using OWL and Jena rules in our prototype.

6 Conclusion

The contribution of this research is a software architecture for inter-organizational multiagent systems. The approach is based on the integration of Web service and Semantic Web technologies into multiagent systems and a uniform transportation of agent and Web service messages.

The resulting combination of technologies enables secure and flexible communication including a virtualization layer for communication endpoints, addressing the requirements of an application in inter-enterprise settings. Furthermore, existing service implementations and infrastructures can be enhanced with technology achievements of the Semantic Web and multiagent areas, enabling complex coordination and adaptation mechanisms to be applied to existing services.

The evaluation has shown that the additional computational effort caused by the uniform communication bus decreases with the number of agents. In addition, we have shown the applicability and utility of our approach in a use case from the airport logistics domain.

Acknowledgments. This work has been supported by the BREIN project (http://www.eu-brein.com) and has been partly funded by the European Commission under contract FP6-034556. We are especially grateful to Joerg Leukel and the participants of the ATOP 2010 workshop for their helpful comments on an earlier version of this paper.

References

1. Bratman, M. E., Israel, J., Pollack, M. E.: Plans and resource-bounded practical reasoning. In: Computional Intelligence 4, 349--355 (1988)
2. Dickinson, I., Wooldridge, M.: Towards Practical Reasoning Agents for the Semantic Web. In: 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03). (2003)
3. FIPA Communicative Act Library Specification, http://www.fipa.org/specs/fipa00037/
4. FIPA Request Interaction Protocol Specification, http://www.fipa.org/specs/fipa00026/
5. Foster, I., Jennings, N. R., Kesselman, C.: Brain meets brawn: why Grid and agents need each other. In: 3rd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), pp. 8--15. (2004)
6. Greenwood D., Calisti, M.: Engineering Web Service - Agent Integration. In: IEEE International Conference on Systems, Man & Cybernetics. IEEE Press, New York (2004)
7. Greenwood, D., Lyell, M., Mallya, A., and Suguri, H.: The IEEE FIPA approach to integrating software agents and web services. In: 6th international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '07). (2007)

8. Haugeneder, H., Steiner, D., McCabe, F.: IMAGINE: A framework for building multi-agent systems. In: Deen, S. M. (ed.) 1994 International Working Conference on Cooperating Knowledge Based Systems (CKBS-94), pp. 31--64. (1994)
9. Howden, N., Rönnquist, R., Hodgson, A., Lucas, A.: JACK intelligent agents - summary of an agent infrastructure. In: 5th International Conference on Autonomous Agents (Agents '01). (2001)
10. IEEE Recommended Practice for Software Requirements Specification, ANSI/IEEE Std 830-1998, IEEE Press, New York (1998)
11. Jade - Java Agent DEvelopment Framework, http://jade.tilab.com/
12. Jade Web Services Integration Gateway (WSIG) Guide, http://jade.tilab.com/doc/tutorials/WSIG_Guide.pdf
13. Jadex BDI agent system, http://jadex.informatik.uni-hamburg.de
14. Jarach, D: The evolution of airport management practices: towards a multi-point, multi-service, marketing-driven firm. Journal of Air Transport Management 7 (2), 119--125 (2001)
15. Karaenke, P., Kirn, S.: A Multi-tier Negotiation Protocol for Logistics Service Chains. In: 18th European Conference on Information Systems (ECIS 2010). (2010)
16. Kipp A, Schubert L, Geuer-Pollmann C. Dynamic Service Encapsulation. In: First International Conference on Cloud Computing. (2009)
17. Laclavík, M., Balogh, Z., Babík, M.: AgentOWL: Semantic Knowledge Model and Agent Architecture. Computing and Informatics 25 (5), 419--437 (2006)
18. Micsik, A., Pallinger, P., Klein, A.: SOAP based message transport for the jade multiagent platform. In: 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Industry track, pp. 101--104. (2009)
19. Moreira, Á.F., Vieira, R., Bordini, R.H., Hübner, J.: Agent-oriented programming with underlying ontological reasoning, In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) Third International Workshop on Declarative Agent Languages and Technologies (DALT-05), pp. 155--170. (2005)
20. Munoz Frutos, H., Kotsiopoulos, I., Vaquero, L. M., Rodero, L.: Enhancing Service Selection by Semantic QoS. In: 6th European Semantic Web Conference on the Semantic Web, pp. 565--577. (2009)
21. Negri, A., Poggi, A., Tomaiuolo, M.: Intelligent Task Composition and Allocation through Agents. In: 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), pp. 255--260. IEEE Press, New York (2005)
22. Nguyen, X. T.: Demonstration of WS2JADE. In: 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05), pp. 135--136. (2005)
23. Paurobally, S., Jennings, N. R.: Protocol engineering for web services conversations. Engineering Applications of Artificial Intelligence 18 (2), 237--254 (2005)
24. Shafiq, O. M., Ali, A., Ahmad, H. F., Suguri, H.: AgentWeb Gateway - a middleware for dynamic integration of Multi Agent System and Web Services Framework. In: 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), pp. 267--270. IEEE Press, New York (2005)
25. Soto, E. L. 2007. Agent Communication Using Web Services, a New FIPA Message Transport Service for Jade. In: P. Petta et al. (eds.): 5th German conference on Multiagent Systems Technologies (MATES 2007), pp. 73--84. (2007)
26. Steiner, D. E., Haugeneder, H., Mahling, D.: Collaboration of knowledge bases via knowledge based collaboration. In: Deen, S. M. (ed.) CKBS-90 — Proceedings of the International Working Conference on Cooperating Knowledge Based Systems, pp. 113--133. Springer, Heidelberg (1991)
27. W3C: Web Ontology Language (OWL), http://www.w3.org/2004/OWL/
28. Wooldridge, M.: Reasoning about rational agents. MIT Press (2000)