Enhancing Multi-Agent Systems with Peer-to-Peer and Service-Oriented Technologies

Marco Mari, Agostino Poggi, Michele Tomaiuolo and Paola Turci
Dipartimento di Ingegneria dell'Informazione,
Università degli Studi di Parma,
Viale G.P. Usberti, 181/A, 43100, Parma, Italy
Tel. +39 0521 905708, Fax +39 0521 905723
{mari, poggi, tomamic, turci}@ce.unipr.it

ABSTRACT
Peer-to-peer and service-oriented technologies have emerged as the dominant means for realizing scalable and interoperable distributed applications. This incontrovertible fact seems to nullify the expectation of multi-agent system researchers that agents could play a fundamental role in realizing such applications. However, from a deeper analysis, it is plain that neither peer-to-peer nor service-oriented technologies can provide by themselves the autonomy and social and proactive capabilities of agents. Motivated by such evidence, several research works have been undertaken with the aim of tackling the problem of integrating peer-to-peer and service-oriented technologies with multi-agent systems. This paper deals with this issue as well. In particular, it shows how the JADE software framework can take advantage of two such technologies both for realizing the infrastructure of distributed multi-agent systems and for supporting the interaction with non-agentized systems.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *multiagent systems*.

General Terms
Management, Performance, Experimentation.

Keywords
peer-to-peer technologies. srvice-oriented technologies, JADE.

1. INTRODUCTION
One of the main challenges of multi-agent systems is to make the realization of scalable distributed systems easy [9][6] and as a consequence to become the main means to support legacy systems interoperability. In the last years, however, two technologies, peer-to-peer and service-oriented, have made an impressive progress and seem to have good chances of competing with multi-agent systems as the main means for the realization of scalable and interoperable systems. Conversely, neither of these two technologies is able to provide by themselves the autonomy and social and proactive capabilities of agents and thus the realization of flexible adaptive distributed systems may be difficult.

Jung, Michel, Ricci & Petta (eds.): *AT2AI-6 Working Notes, From Agent Theory to Agent Implementation, 6th Int. Workshop,* May 13, 2008, AAMAS 2008, Estoril, Portugal, EU.

Confirming what has been said above, most people involved in multi-agent systems research are firmly convinced that an integration of multi-agent systems with the aforementioned technologies seems to be the most suitable solution for the realization of scalable and interoperable distributed applications. As a matter of fact, in the last years a lot of work has been presented for the integration of multi agent systems with one or both of the two technologies [24][12][13][4].

This paper is situated in this context. In particular, it shows how JADE, one of the best known and most used software framework for the development of multi-agent systems [2][14], has been extended with these technologies both to support the realization of multi-agent systems and to facilitate the interoperability with peer-to-peer and service-oriented systems. The next two sections respectively describe the peer-to-peer and the service-oriented extensions of the JADE software framework. Section 4 presents a JADE multi-agent system that has been realized exploiting the extensions of JADE and that supports collaboration via an information and expert finder service. Finally, section 6 summarizes the contributions of our work and points out future lines of research.

2. EXTENDING JADE WITH PEER-TO-PEER TECHNOLOGIES
The traditional, client-server model describes systems where computational resources and data are centralized in few servers, which respond to requests of clients. On the other hand, clients are supposed to have little capabilities and rely on the resources of servers for most of their tasks. The multi-agent model reverses this paradigm and describes systems organized in a peer-to-peer fashion, where each participant potentially has some resources to share and some services to offer to the community of agents. Thus, according to the context, each agent is able to play either the role of client or server.

JADE implements FIPA specifications for multi-agent systems, and so enables the realization of peer-to-peer distributed systems, constituted by smart and loosely coupled agents communicating by means of asynchronous ACL messages.

Nevertheless, JADE does not exploit some important features of modern peer-to-peer networks, in particular:

1. The possibility to build an "overlay network", hiding differences in lower level technologies and their related communication problems;

2. The possibility to build a completely distributed, global index of resources and services, without relying on any centralized entity.

Some multi-agent systems, like Agentscape, approached the same issues by developing a dedicated peer-to-peer network layer [17]. For JADE, we choose to integrate agent platforms into an already existing and used peer-to-peer environment like JXTA, thus, benefiting from a well tested system and exposing services to other entities participating in the networkr.

2.1 JXTA-MTP

In the course of some large projects based on agent technologies like Agentcities and @lis TechNet [18][1], some recurring problems emerged at the level of connection among remote platforms. The importance of these problems invariably grows with the cardinality and geographical extension of the interconnected infrastructure, and has been acknowledged in other similar large scale environments.

Most peer-to-peer networks specifically address this kind of problems allowing the connection of peers located behind firewalls, Network Address Translators (NATs) and Dynamic Host Configuration Protocol (DHCP) servers, or requiring different and particular protocols like HTTP or WAP. To this end, peer-to-peer networks create an overlay infrastructure above underlying diverse and problematic links in order to realize a more abstract and homogeneous ground and simplify the communications among peers.

One of the most used technologies for this purpose is JXTA [15]. JXTA technology is a set of open, general-purpose protocols that allows any connected device on the network (from cell phones to laptops and servers) to communicate and collaborate in a peer-to-peer fashion. The project was originally started by Sun Microsystems, but its development was kept open from the very beginning. JXTA comprises six protocols allowing the discovery, organization, monitoring and communication between peers. These protocols are all implemented on the basis of an underlying messaging layer, which binds the JXTA protocols to different network transports.

JXTA peers can form peer groups, which are virtual networks where any peer can seamlessly interact with other peers and resources, whether they are connected directly or through intermediate proxies. JXTA defines a communication language which is much more abstract than any other peer-to-peer protocol, allowing to use the network for a great variety of services and devices. A great advantage of JXTA derives from the use of XML language to represent, through structured documents, named *advertisements*, the resources available in the network. XML adapts without particular problems to any transport mean and it is already an affirmed standard, with good support in very different environments, to structure generic data in a form easily analyzable by both humans and machines.

With respect to connectivity, JXTA does not suppose a direct connection is available between all couple of peers. Peers can use the *Peer Endpoint Protocol* to discover available routes for sending a message to a destination peer. Particular peers, called *routers*, are in charge of responding to such queries providing route information, i.e. a list of *gateways* connecting the sender to the intended receiver. A gateway acts as a communication relay, where messages can be stored and later collected by their intended recipient, overcoming problems related to limited connectivity.

JADE, on the other hand, offers an extensible mechanism for the transport of messages among platforms, in the form of pluggable Message Transport Protocols (MTPs). The default implementations are based on IIOP and HTTP, which are both limited by the requirement of a direct connection between sender and receiver.

Exploiting the extensibility of JADE platforms, a JXTA-MTP has been developed at the University of Parma, which overcomes these limitations. To transport messages between two platforms, the new MTP uses JXTA pipes which are bound to specific endpoints (typically an IP address and a TCP port) only dynamically. JXTA pipes are advertised on the network in the same way as other services offered by peers, and provide a global scope to peer connectivity.

The JXTA-MTP implementation allows using not only plain JXTA pipes, but also secure ones with encryption and signature mechanisms guaranteeing privacy, integrity and authenticity of exchanged messages.

2.2 JXTA-ADS

What usually happens in a multi-agent platform is the cohabitation of multiple agents interacting in a common and cohesive environment, making use of a formal communication language defined by its own syntax and semantics, in order to complete tasks demanded by users. For the communication to be constructive, it is necessary to provide agents with a system allowing them to reciprocally individuate offered services. This happens thanks to the presence of a yellow pages service, provided by the platform, which can be consulted by agents when needed. However this often limits the search inside a single platform. Solutions are possible, which allow the consultation of other yellow pages services, but they necessitate the a priori knowledge of the address of the remote platform where services are hosted or listed.

An alternative solution is represented by a yellow pages service leaning on a peer-to-peer network like JXTA, thanks to which each network device is able to individuate in a dynamic way services and resources of other network devices.

Technologies inherent to web services are imposing WSDL as a standard language to publicize all different available resources. In FIPA, a simpler formalism is defined to describe services and resources exposed by agents and linked to their own domain ontology. JXTA does not establish any constraint on the way to describe and invoke services. JXTA protocols simply provide a generic framework, allowing the use of any mechanism, also WSDL or FIPA service descriptions, to exchange information needed to invoke a service.

Particular peers, called *rendezvous* peers, are in charge of indexing resources made available in the network and find them when requested by other peers. Rendezvous peers can also communicate queries to each other, if they do not possess the right information, thus enabling the discovery of advertisements beyond the local network.

In fact, in JXTA, resources are described by advertisements, which are essentially XML documents collecting metadata of available resources. Advertisements are not stored on some single machine, such as a server, or on a hierarchical infrastructure. They are distributed among rendezvous peers, which implement a distributed algorithm, called *shared resource distributed index* (SRDI), for the creation and management of the index of resources available in the network. On the basis of some indexed attributes, the mechanism can solve queries made anywhere in the rendezvous network. Basically, the global index is a loosely consistent distributed hash table, where the hash of an indexed attribute is mapped to some peer responsible for storing the actual advertisement.

FIPA has acknowledged the growing importance of the JXTA protocols, and it has released some specifications for the interoperability of FIPA platforms connected to peer-to-peer networks. In particular, in [7] a set of new components and protocols are described, to allow the implementation of a DF-like service on a JXTA network. These include:

Generic Discovery Service – a local directory facilitator, taking part in the peer-to-peer network and implementing the Agent Discovery Service specifications to discover agents and services deployed on remote FIPA platforms working together in a peer-to-peer network.

Agent Peer Group – a child of the JXTA Net Peer Group that must be joined by each distributed discovery service.

Generic Discovery Advertisements – to handle agent or service descriptions, for example FIPA df-agent-descriptions.

Generic Discovery Protocol – to enable the interaction of discovery services on different agent platforms. It's a request/response protocol to discover advertisements, based on two simple messages, one for queries and one for responses.

The JADE development environment does not provide any support for the deployment of real peer-to-peer systems because it only provides the possibility of federating different agent platforms through a hierarchical organization of the platform directory facilitators on the basis of a priori knowledge of the agent platforms addresses. Therefore, at University of Parma the JADE directory facilitator has been extended to realize a peer-to-peer network of agent platforms thanks to the JXTA technology [15] and thanks to two preliminary FIPA specifications for the Agent Discovery Service [7] and for the JXTA Discovery Middleware [8].

This way, JADE integrates a JXTA-based Agent Discovery Service (ADS), which has been developed in the respect of relevant FIPA specifications to implement a GDS. Each JADE platform connects to the Agent Peer Group, as well as to other system-specific peer groups. The Generic Discovery Protocol is finally used to advertise and discover agent descriptions, wrapped in Generic Discovery Advertisements, in order to implement a DF service, which in the background is spanned over a whole peer group.

3. EXTENDING JADE WITH SERVICE-ORIENTED TECHNOLOGIES

Industry is increasingly interested in executing business functions that span multiple applications, thus requiring high-levels of interoperability and a more flexible and adaptive business process management. The most appropriate response to this need seems to be having systems assembled from a loosely coupled collection of Web services. This technical area appears to be an interesting environment in which the agent technology can be exploited with significant advantages. As a matter of fact, several researches belonging to the agent community have dealt with the issues concerning the interconnection of agent systems with W3C compliant Web services, with the aim of allowing each technology to discover and invoke instances of the other. One evident benefit of this is the central role that agents could play in a service-oriented scenario, by efficiently supporting distributed computing [3] and allowing the dynamic composition of Web services.

The proposed integration approaches [11][16][19] denote different shades of meaning of the same idea, i.e. a wrapper or an adapter module playing the role of mediator between the two technologies. Most of them have adopted the gateway approach, providing a translation of WSDL descriptions and UDDI entries to and from FIPA specifications, thereby limiting the communication to simple request-response interactions. One approach [21], which differentiates quite substantially from the others, realizes a FIPA compliant JADE Message Transport System for Web Services enabling agents to interact through the Web with Web services preserving the FIPA compliant communication framework. But, as stated by the author himself, it only provides a solution for an integration at a low level, leaving a number of issues at higher levels unresolved.

These efforts towards an integration of the two technologies, which imply a mapping between two different ways of thinking about communication patterns, are in our opinion appreciable but at the same time quite arguable or at least unnecessarily complex.

In the following, we try to explain our point of view, analyzing and comparing the two technologies.

As far as FIPA is concerned, we can assert that the inter-agent communication is dealt with in several documents and definitely represents an important part of the overall specifications. In our attempt to be concise and rigorous, we can state that FIPA specifications target autonomous agents expected to communicate at a high level of discourse, whose contents are meaningful statements about agents' knowledge and environment. The FIPA Agent Communication Language is based on the speech act theory; messages are communicative acts that, by virtue of being sent, have effects on the knowledge and environment of the receiver as well as the sender agent. Furthermore, the language is described using formal semantics based on the modal logic. From this it clearly emerges the communication complexity which characterizes multi-agent systems compared to the very simple conversation patterns of the Web services.

It is only fair to say that Web services are also suitable for high-level communication patterns and that in the last years several efforts have been carried out in order to provide description languages enabling service orchestration and choreography and

moreover to make Web services semantically described. Nevertheless, the main goal is still to improve interoperability between entities that are not necessarily characterized by sophisticated reasoning capabilities. Finally, it is important to highlight that, in spite of the efforts dedicated to maintaining the service session, the nature of Web services is almost stateless in contrast with agents that are in essence stateful.

To sum up, the two entities, Web services on the one hand and agents on the other hand, and the corresponding communication patterns are very different from a semantic point of view. That raises doubts about a possible mapping between the two worlds, for two main reasons: (i) the essential differences between the two communication patterns, which imply a loss of descriptive power in the mapping and (ii) the unnecessary overhead that this mapping inevitably causes.

3.1 The Proposed Solution

Bearing in mind what said above, we have implemented a framework which allows a single agent to directly communicate with the world of Web services without passing through the FIPA ACL messages. When the agent needs to invoke a Web service it creates, by using our framework, the SOAP message and sends it to the provider; likewise the agent can provide its services as Web services. For the implementation of our framework we exploited AXIS2 as it is one of the most updated implementations of the Web Services standards. We have followed a similar approach also in the case of the publishing and discovery of services.

At this point a question arises: are these JADE agents still FIPA compliant? In our opinion, they are still FIPA agents since the interaction with other agents is carried out according to FIPA specifications whilst when the agent wants to communicate with the Web service world it conforms itself to the Web services standard.

Our research work does not just provide a “syntactic” support suitable for interactions with Web services compliant with the basic profile, i.e. WS-I Basic Profile 2.0, but we have also tried to cope with the issues related to a semantic support. Even though we were aware that Web services supplied by different providers usually have individual and unique semantics, described by independently developed ontologies, in our attempt towards a semantic support, we consider the simplified, but still significant, case of a shared ontology that gives a common knowledge background to the entities in the system. In order to facilitate the resolution of structural and semantic heterogeneities, semantic Web services have their interfaces semantically described by ontological concepts belonging from this shared ontology.

In the case of semantic Web services, it is appropriate to give the agent support in:

Looking for a service on the basis of the requirements to be met by the service itself;

Service invocation requiring simply the input data, irrespective of the data format and the way of interacting with the real service.

One of the major issues, we dealt with, concerned the mapping between the semantic description of the service and its real invocation. That is, how to deduce, starting from the abstract service description, which refers to ontological concepts, the concrete data required to invoke the service. The way in which this mapping is done heavily depends on the specifications chosen for the incorporation of machine understandable semantics.

Several proposals have been submitted to W3C in order to make Web services semantically described. From an analysis of such submissions two possible alternatives emerge:

The definition of a service ontology (domain-independent), to which one has to refer for a semantic description of the service. Such a description is in correlation with the WSDL document of the real service;

The definition of specifications to semantically annotate the service WSDL document. These annotations associate elements belonging to the WSDL document with concepts belonging to domain ontologies.

Both alternatives are based on the reference to one or more domain ontologies, in which the concepts, referred in the semantic part of the service description, are defined.

Among the second group, one became a recommendation last year, i.e. SAWSDL. From the first group, one is quite interesting, even if not a recommendation yet. It is a proposal submitted by a community of researchers and it is about an ontology of service, called OWL-S, characterized by a more comprehensive approach to the semantic orientation of the Web service description.

When we started our research work, SAWSDL was not a recommendation yet and OWL-S was the most visible of the several proposals. The semantic expressivity of OWL-S is rich and quite flexible. It defines a new way to describe Web service profile and so it slightly overlaps the content of the WSDL document. If one adopts OWL-S as a language to semantically describe Web services, it is necessary to handle two documents: WSDL, mainly for binding information; OWL-S, for semantic references.

Our initial choice fell down on OWL-S. Now we are extending the framework in order to include a support for the SAWSDL specification.

In order to allow agents to be able to produce and consume semantically annotated information and services, it is necessary to provide them with an ontology management support.

Ontologies were considered by the FIPA community too. In fact, ontologies enable agents to communicate in a semantic way, exchanging messages which convey information according to explicit domain ontologies. FIPA specifications, however, do not state anything about neither how to utilize ontology in the message content nor the ontology language to use.

As far as JADE is concerned, the idea which mostly inspired the design of the JADE content language and ontological support was to define an ontology independent abstract model of the content language that could be subsequently bound to any domain ontology representation expressed using an object-oriented data model. This ontological support has been conceived when the Semantic Web was on its very early stage of research and development and OWL was not already established as a standard. Consequently its expressive power is clearly limited with respect to OWL and basically allows expressing taxonomy of concepts, predicate and actions and therefore it is not able to represent completely the different application domains where JADE agent

may be used. In order to provide a JADE agent with an adequate expressive power, it is necessary either to replace or to integrate the JADE ontological support.

In the attempt to find a suitable solution to this problem in a previous work we realized a tool called OWLBeans [23]. OWLBeans, conceived with the goal of providing simple artefacts to access structured information, represents a light support allowing agents to import OWL ontologies as an object-oriented hierarchy of classes. It clearly shows a limited expressive power but still sufficient in the first phase of our development in which we focused only on a hierarchical representation of concepts for the description of input and output service parameters. In this specific context, as a matter of fact, agents do not need to face the computational complexity of performing inferences on large, distributed information sources, but an object-oriented view of the application domain is enough to allow them to complete the tasks of publishing, discovery and invocation of semantic Web services.

Well aware that the implemented framework represents a first step towards the interoperability between agents and semantic Web services, we have already starting working on the realization of a full OWL DL support supplying ontology management and reasoning functionalities, with the main purpose of providing a more expressive and powerful support and in the meantime of reducing the amount of computational resources and time required (compared to the Jena engine).

4. AN INFORMATION AND EXPERT FINDER MULTI-AGENT SYSTEM

RAP (Remote Assistant for Programmers) is a system to support communities of students and programmers during shared and personal projects based on the use of the Java programming language. RAP associates a Personal Agent with each user which helps her/him to solve problems, proposing information and answers extracted from some information repositories and forwarding answers received by "experts" on the topic selected on the basis of their profile.

4.1 System Agents

The system is mainly based on three kinds of agents: Personal Agents, Data Miner Agents and Rate Evaluator Agents.

Personal Agents (PAs) allow the interaction between a user and the different parts of the system and, in particular, between the users themselves. Moreover, these agents are responsible for building the user profile and maintaining it when the user is "on-line". User-agent interaction can be performed in two different ways: when the user is active in the system, through a Web based interface; when it is "off-line" through emails. Usually, there is a Personal Agent for each on-line user, but, when needed, Personal Agents are created to interact with "off-line" users via emails.

Data Miner Agents (DMAs) are responsible for maintaining system documentation and finding the appropriate "pieces of information" to answer the queries submitted by the users. The documentation is composed by three elements: i) code (e.g. Java classes), ii) a repository of all answers provided during system life and iii) a repository of relevant documents submitted by the users (e.g.: tutorials, manuals, presentations).

Rate Evaluator Agents (REAs) perform the calculation of users expertise score and of documents relevance with regard to the answers posed by system users. REAs are closely linked to DMAs, from which they take data for the calculations. The role of REAs is fundamental for achieving the distributed, peer-to-peer nature of the system, as discussed in section 4.4.

Each RAP platform obviously hosts a Directory Facilitator agent. Such agent not only provides the yellow pages service for its platform agents, but integrates the JXTA-based Agent Discovery Service presented in section 2.2. In this way, several RAP communities can be connected as elements of a peer-to-peer network.

4.2 System Behavior at a Glance

The system architecture is quite complex and a complete description of RAP features is beyond the scope of this paper. However, it is possible to outline the RAP behavior showing the scenario of a user asking information to her/his Personal Agent to solve a problem in her/his code. The description of this scenario can be divided in the following steps:

1. Select answer types
2. Submit a query
3. Find answers
4. Rate answer

Select answer types: the user can choose the source for the information to receive among code repositories, system documentation, old answers repositories and new answers sent by other system users.

Submit a query: the user provides the query to her/his Personal Agent. In particular, the user can query either about a class or an aggregation of classes for implementing a particular task or about a problem related to her/his current implementation.

Find answers: the Personal Agent interacts with Rate Evaluator Agents to collect the required answers. If the user requested answers from other system users, the activity is more complex and its description can be divided in three further steps:

3.1) **Receive experts rating**: a rating to answer the query is calculated for each user on the basis of her/his profile. The identity and a sketched profile of each user with a positive rating is forwarded to the requesting Personal Agent.

3.2) **Select experts**: the Personal Agent divides on-line and off-line users, orders them on the basis of their rating and, finally, presents these two lists to its user. The user can select one or more recipients for her/his query. On-line users will receive the query immediately (through their PAs), off-line users via e-mail and as they will connect to the system.

3.3) **Receive answers**: selected "expert" users can answer the query in the system Web interface or through an e-mail. Provided answers are presented to the querying user as soon as they arrive.

Rate answers: after the reception of all the answers (from every requested source), or when the deadline for sending them expired, or, finally, when the user already found an answer satisfying her/his request, the Personal Agent presents the list of read answers to its user asking her/him to rate them. Each rating is

forwarded to a Data Miner Agent that updates the involved profiles.

4.3 User and Document Profile Management

Profiles are represented by vectors of weighted terms whose values are related to the frequency of the term in the document or to the term frequency in the code wrote by the user. Document and user profiles are computed by using term frequency inverse document frequency (TF-IDF) [20] and profiles weighted terms correspond to the TF-IDF weight. Some problems have risen applying this approach in a multi-platform and distributed system: we present these problems and discuss the solutions in the next section. Each user profile is built by user's Personal Agent through the analysis of the Java code she/he has written. The profile built by Personal Agents is only the initial user's profile, and it will be updated when the user submits new software she/he has written and when the user helps other users answering their queries.

4.4 Open and Distributed Communities

An important requirement that guided the design of RAP was the support for open and distributed users communities. In fact, the retrieving of experts and information can be improved if the communities beneath the system have the capability to grow including new users or new communities.

RAP structure is open, because new users can register and access the system, but also because a registered user can acquire new skills or write new code and therefore update her/his profile. Obviously, it's also possible to delete a user.

The community beneath RAP is distributed because the whole system can consist of a dynamic group of local communities. Each community can exist and operate isolated, but can also decide to join a group of communities, sharing the experts and the document repositories. The joining and the leaving of a community are dynamic operations, the single communities of a group are fully independent, just like the components of a peer-to-peer network. A community has the capability to discover and federate with other communities thanks to the enhanced Directory Facilitator presented in section 2.2, therefore without a previous knowledge of other communities platform address.

The open and distributed nature of the system provides the best conditions for sharing and retrieving information, but also entails some significant problems in the evaluation of such information. In fact, the evaluation of both experts and documents is strongly dependent on the actual composition of the community group. For example, if a user is rated the maximum expert to answer a query, he is rated considering only the users registered in the system at the moment of the rating. If a new local community joins the group, it is possible that a user with more experience has become available. In this case, an information like "user A has called n times a method of the class X" is still valid, but an information such as "user A is the maximum expert in class X" may change according to the composition of the community.

The problem rises from the fact that, while the user personal information is still valid, the rating and all other information related to the community must be recalculated. As a matter of fact, TF-IDF algorithm can be easily used in a centralized system where all the profiles and the data to build them are managed. Our context is more complex: the system is distributed, only the Personal Agent can access the software of its user, for privacy and security reasons, and the profiles are maintained by the interaction of Personal Agents and Data Miner Agents.

For these reasons, each profile component of the RAP system is associated with two elements: an absolute element and a TF-IDF weighted element. The absolute one is dependent only on the user (or document) profile. The TF-IDF weighted element is dependent on both the user profile and the whole community profiles. While the absolute element is stored in a database, the weighted one is maintained in memory and it is recalculated when necessary. Obviously, every rating is determined on the basis of the weighted element.

The situations in which could be necessary to recompute the weighted element of one or more profile components can be slightly frequent:

- A new community joins or leaves the community group;
- A new user registers or is deleted from the system;
- Some components of a user profile change: for example the user submits new software or receives a rating for an answer.

For performance reasons, particularly if the community beneath the system is large, the process of recalculation could be too expensive in terms of system resources. In this case, each RAP platform administrator can force the system to perform the necessary recalculations only at a scheduled time.

4.5 External Sources of Information

While RAP information repositories represent a complete source of information when the query is about a class of the code repository, or about a problem already faced by other system users, clearly they lack information about a wide range of programming problems. In this sense, we are planning to open the system to external, possibly heterogeneous, sources of information. In a service-oriented scenario, the natural choice for such sources to provide their content is through a Web Service. For example, Google provides a complete set of SOAP APIs [10] to query its search engine, or Systinet [22] provides a W3C Search Service to search over a repository of tutorials and references covering most XML languages.

Exploiting the framework presented in section 3, we developed and integrated in RAP a Web Service Agent. This agent has the task to receive a query from a Personal Agent, create SOAP messages, send them to a group of registered Web Services and forward the results to the requesting Personal Agent. At the moment, the results are not rated by a Rate Evaluator Agent, but if the resource provided by a Web Service is rated useful by a user, the resource link is registered in the answer repository of the system.

5. CONCLUSION

This paper has dealt with the issue of enhancing the multi-agent systems role in the realization of scalable and interoperable systems by exploiting peer-to-peer and service-oriented technologies as key components for their realization. In particular, the paper has shown how JADE, one of the best known and most used software framework for the development of multi-agent

systems, has been extended with these two technologies both to support the realization of multi-agent systems and to facilitate the interoperability with peer-to-peer and service-oriented systems.

The resulting system shows interesting advantages. On the one hand, the exploitation of the peer-to-peer technology gives a great impulse towards the scalability and interoperability of different agent platforms and agent-based applications. On the other hand the openness towards the Web service standards gives agents the possibility to interoperate with a consolidated industrial reality and one of the most accepted mechanisms used for integration of distributed systems.

Additional advantages can be gained by coupling multi-agent systems with Semantic Web techniques [5]. In fact, agents could play a central role in a service-oriented scenario, by efficiently supporting distributed computing and allowing an automatic and intelligent composition of Web services. Furthermore they may also be the means for harmonising and making straightforward the interoperability between the services provided by peer-to-peer, service-oriented and multi-agent systems. Our future work will be oriented towards the enhancement of the JADE software framework by extending the current ontology support with Semantic Web techniques (i.e., use of OWL and related reasoning techniques) and definition and experimentation of a shared format for the publication of peer-to-peer, service-oriented and multi-agent systems services.

6. REFERENCES

[1] @lis TechNet Web Site (2008). Available from: http://www.alis-technet.org/.

[2] Bellifemine, F., Poggi, A., Rimassa, G. Developing multi agent systems with a FIPA-compliant agent framework. in Software - Practice & Experience, 31:103-128 (2001).

[3] Bergenti, F., Rimassa, G., Somacher, M., Botelho, L.M. A FIPA Compliant Goal Delegation Protocol. Communication in Multiagent Systems, Vol. 2650, pp. 223-238. 2003. Springer

[4] Buford, J. and Burg, B. Using FIPA Agents with Service-Oriented Peer-to-Peer Middleware. In Proc. of the 7th Int. Conf. on Mobile Data Management, Nara, Japan (2006).

[5] Burstein, M.H., Bussler, C., Zaremba, M., Finin, T.W., Huhns, M.N., Paolucci, M., Sheth, A.P., Williams, S.K. A Semantic Web Services Architecture. IEEE Internet Computing 9(5):72-81 (2005).

[6] FIPA Specifications Web Site . Available from http://www.fipa.org.

[7] FIPA Agent Discovery Service Specification. 2003. Available from http://www.fipa.org/specs/fipa00095/PC00095.pdf.

[8] FIPA JXTA Discovery Middleware Specification. 2004. Available from http://www.fipa.org/specs/fipa00096/PC00096A.pdf.

[9] Genesereth, M.R. An agent-based framework for interoperability. In Software Agents, J. M. Bradshaw, pp. 317-345, Ed. MIT Press, Cambridge, MA (1997).

[10] Google SOAP API home page. Available at: http://code.google.com/apis/soapsearch/index.html

[11] Greenwood D. and Calisti M. Engineering Web Service-Agent Integration. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pages 1918–1925, The Hague, Netherlands (2004)

[12] Greenwood, D., Nagy, J., Calisti, M. Semantic Enhancement of a Web Service Integration Gateway, Proc. of the AAMAS 2005 workshop on Service Oriented Computing and Agent Based Engineering (SOCABE), Utrecht, Netherlands (2005).

[13] Huhns, M.N., Singh, M.P., Mark H. M.H., Decker, K.S., Durfee, E.H., Finin, T.W., Gasser, l., Goradia, H.J., Jennings, N.R., Lakkaraju, K., Nakashima, H., Parunak, K., Rosenschein, J.S., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K.P., Tambe, M., Wagner, T., Zavala Gutierrez, R.L. Research Directions for Service-Oriented Multiagent Systems. IEEE Internet Computing 9(6):65-70 (2005).

[14] JADE Web Site (2008). Available from: http://jade.tilab.com/.

[15] JXTA Web Site (2008). Available from: http://www.jxta.org/.

[16] Nguyen X. T. Demonstration of WS2JADE. In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pages 135–136, Utrecht, The Netherlands, (2005).

[17] Overeinder, B.J., Posthumus, E., Brazier, F.M.T. Integrating Peer-to-Peer Networking and Computing in the AgentScape Framework. In Proc. Second International Conference on Peer-to-Peer Computing, P2P02, Linköping, Sweden (2002).

[18] Poggi, A., Tomaiuolo, M., Turci, P. Using agent platforms for service composition. In. Proc. 6th International Conference on Enterprise Information Systems (ICEIS-2004), pp. 98-105, Porto, Portugal, 2004.

[19] Shafiq M. O., Ali A., Ahmad H. F., Suguri H. AgentWeb Gateway - a Middleware for Dynamic Integration of Multi Agent System and Web Services Framework. In Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pages 267–270, Washington, DC, (2005)

[20] Salton, G.: Automatic Text Processing. (1989), Addison-Wesley

[21] Soto E. L. Fipa agent messaging grounded on web services. In Proceedings of the 3rd International Conference on Grid Service Engineering and Management, (2006)

[22] Systinet Home Page. Available at: http://www.systinet.com

[23] Tomaiuolo, M., Turci, P., Bergenti, F., Poggi, A., An Ontology Support for Semantic Aware Agents. In Proc. Seventh International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2005 @ AAMAS), Utrecht, The Netherlands, (2005)

[24] Willmott, S., Pujol, J.P., Cortés, U. On Exploiting Agent Technology in the Design of Peer-to-Peer Applications. Proc. of the 3rd International Workshop on Agents and Peer-to-Peer Computing, pp. 98-107, New York, NY (2004)