An Agent-Based Framework for Integrating Mobility into Grid Services

T.E. Athanaileas
School of Electrical and Computer Engineering, National Technical University of Athens
9 Heroon Polytechneiou Str.
15773, Zografou, Athens, Greece
thathan@esd.ntua.gr

N.D. Tselikas
School of Electrical and Computer Engineering, National Technical University of Athens
9 Heroon Polytechneiou Str.
15773, Zografou, Athens, Greece
ntsel@telecom.ntua.gr

G.V. Tsoulos
Department of Telecommunications Science and Technology, University of Peloponnese
Tripolis 22100, Greece
gtsoulos@uop.gr

D.I. Kaklamani
School of Electrical and Computer Engineering, National Technical University of Athens
9 Heroon Polytechneiou Str.
15773, Zografou, Athens, Greece
dkaklam@mail.ntua.gr

ABSTRACT
This paper presents an approach to integrating mobility into the originally static grid environment. The mobile agent paradigm has been exploited in order to facilitate the migration of grid services. The approach presented here separates each mobile grid service in a static part and a mobile part that can dynamically migrate across grid nodes as appropriate. In order to enable this approach, a framework based on a novel mobile agent system is proposed. The components of the framework consist exclusively of grid services conforming to the Web Services Resource Framework (WSRF) standards. The framework provides mechanisms to seamlessly integrate mobile agents and stateful grid services.

Categories and Subject Descriptors
D.2.11 [**Software Engineering**]: Software Architectures –*domain specific architectures, patterns.*

General Terms
Design

Keywords
Grid services, mobile agents, WSRF, mobile services

1. INTRODUCTION
Grid Computing is an emerging computing model targeting to distributed processing across a parallel infrastructure and is distinguished from conventional distributed systems by its focus on large-scale resource sharing applications and its high performance orientation [1]. The Grid problem is defined as the problem of secure, flexible and coordinated resource-sharing across multiple collections of individuals, systems or resources, which are referred to as virtual organizations.

Two central issues in the development of such large scale systems is the achievement of interoperability between a wide collection of resources and components that can be extremely heterogeneous, as well as the openness of the architecture and the protocols that are used. Towards this direction, a service-oriented architecture has been widely accepted as the basis for the composition of grid infrastructures, namely the Open Grid Services Architecture - OGSA [2]. According to OGSA, a grid is considered to be a collection of web services, which use standardized protocols for description, discovery and invocation. Recently, the notion of state has been added to the originally stateless web services, resulting in the creation of stateful web services and the Web Services Resource Framework (WSRF [3]) has been proposed as standardization for interaction with stateful resources.

An OGSA based grid as described above has the limitation that the provided services can be only of a stationary nature. Mobility is a characteristic not yet supported, although there are certain advantages in considering services that have the ability to move among nodes. Characteristics such as optimized resource utilization, improved coordination between services and available resources and overall framework flexibility can be propelled by integrating mobility to grid services.

In order to attain mobility integration, a grid service mobility framework based on the paradigm of mobile agents is proposed in this paper. The purpose of the presented work is to enable the combination of the functionality and characteristics provided by the agent paradigm with the standardization provided for grid services, towards creating autonomous services, richer service models and semantic integration.

The outline of the paper is as follows: in Section 2 related work is discussed. In Section 4, the agent system that is used as a basis for mobility integration is concisely presented. Section 5 presents the details of the grid service mobility framework. Finally, Section 6 concludes the paper and presents some goals for future work.

2. RELATED WORK
The work presented in this paper has been strongly influenced by several research contributions in the area of integrating mobile

Mobilware '08, February 12-15, 2008, Innsbruck, Austria.

agents, web services and grid computing. In [4] and [5], a comparison between web services and agents has been made, concerning the transport layer and the service discovery protocols. Additionally, certain challenges for manipulating agents as standard web services have been addressed to some point. In [6], an agent-based approach to composite mobile web services has been presented. According to this approach, web service composition is accomplished by having an agent perform the composition on behalf of a mobile user, while interactions between web services are transparent to the user. In [7], an integration infrastructure has been developed, with the twofold goal of exporting agent functions with a web service interface and simplifying the access to legacy web service components from any agent system, thus promoting interoperability between the two technologies. In [8], a framework for integrating multi agent systems with the grid service architecture has been proposed, with the ultimate goal of creating an autonomous semantic grid. Recently, in [9], a middleware framework for secure mobile grid services has been discussed. This framework makes use of the Jade agent platform and the Globus Toolkit 4 functionality to facilitate the creation of mobile grid services.

The motivation behind combining grid systems and the agent computational paradigm has been discussed in [10]. Although grid systems provide powerful mechanisms for secure and reliable resource sharing, they can in general be considered to be rigid and inflexible in terms of their interoperation and their interactions. On the other side, research on agents has led to the development of concepts, methodologies and algorithms for autonomous entities that can be extremely flexible in dynamically varying conditions. A thorough discussion concerning various and different ways grids and agents can benefit from each other is cited in [10].

In the work presented in this paper, we aim at building on some of the concepts previously described, towards incorporating mobility to an OGSA-based grid infrastructure. In order to achieve such a combination between grid services and agent paradigm, the first logical step would be to create a common run-time for the components of the two technologies, as well as standardized interfaces for accessing them. The work presented in the following sections moves towards that direction.

3. THE GRID SERVICES BASED AGENT PLATFORM

The Mobility Integration Framework is based on our proposed Grid Services Based Agent Platform (GriSBAP). GriSBAP is an agent framework constructed entirely as a collection of stateful web services. It acts as a run-time environment enabling the creation of agents and supporting agent life-cycle. Its structure and operations partially (for the moment) conform to the FIPA specifications [11], in order to facilitate interoperability with other systems. GriSBAP has been implemented using Java and the functionality provided by the Globus Toolkit 4 Java WS Core. The Java WS Core is part of the Globus Toolkit [12], which has been developed by the Globus Alliance [13] as a grid-enabling middleware. The Java WS Core provides the necessary functionality for the creation of stateful web services that conform to WSRF standardization.

The main advantages of using a grid service based agent framework instead of importing existing agent frameworks to the grid environment are the minimization of overheads as well as the homogeneity with the underlying grid environment. The following paragraphs shortly present the structure and the functionality of the agent framework.

3.1 Agent Platform Structure

The structure of GriSBAP is depicted in Figure 1.

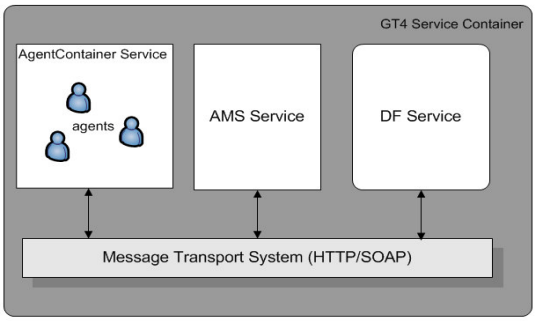

Figure 1. Agent Platform Structure

As proposed by FIPA Agent Management Specification [14], the agent platform consists of the following entities:

- *The AMS Service*: the AMS (Agent Management System) represents the authority entity in the platform. The AMS component has the ability to perform various actions on the agents' life-cycle (e.g. create or kill agents) and also provides the naming service. Each agent platform necessarily contains one single instance of the AMS component.
- The DF Service: the DF (Directory Facilitator) provides a yellow pages directory service to agents. Any agent can publish its functionality and its services offered by registering to the DF, while an agent that needs to use a specific service can query the DF in order to discover an agent that provides this service.
- *The AgentContainer Service*: this component is the actual run-time environment for an agent. The AgentContainer provides functionality regarding agents' life-cycle management, which can be used from the AMS to manage the agent system.
- *The Message Transport System*: it constitutes the communication bus of the platform. The communication between agents is achieved by exchanging ACL (Agent Communication Language) messages. This kind of communication is obviously a rather abstract way of communication and the actual exchange of messages has to be made via means of a concrete communication protocol. The message transport system of the platform makes use of the SOAP [15] protocol; each ACL message is encapsulated into a SOAP message and sent over HTTP.

All platform components are implemented as stateful web services conforming to WSRF standards. Their functionality is exposed by standard WSDL [16] interfaces and the invocation of the exposed operations is made by standard SOAP requests and responses. Additionally, all components publish resource properties that are representative of their state. The AMS Service publishes two

kinds of resources: (a) the platform name and (b) the structure of the platform, which is a list of the URIs of the instances of the AgentContainer Service that have registered with the specific AMS, as well as the identifiers of the agents that reside in each instance. The resource properties of the AgentContainer Service are (a) the name of the AgentContainer and (b) a list of the agent identifiers that currently reside in the AgentContainer. Finally, the DF Service exposes as a single resource property the list of agents that have been registered with it and a description of the services they provide.

The monitoring functions of the framework (such as agent state monitoring) are performed by using the WS-Notification [17] implementation of the Globus Toolkit 4. The AgentContainer Service implements the NotificationProducer portType of the WS-Notification standard, while the AMS Service acts as a notification consumer, being notified about the changes in the AgentContainer resource properties. The DF Service also implements the NotificationProducer portType, so that an agent can be notified about services provided by other agents.

All platform components are implemented as stateful web services conforming to WSRF standards. Their functionality is exposed by standard WSDL [16] interfaces and the invocation of the exposed operations is made by standard SOAP requests and responses. Additionally, all components publish resource properties that are representative to their state. The AMS Service publishes two kinds of resources: (a) the platform name and (b) the structure of the platform, which is a list of the URIs of the instances of the AgentContainer Service that have registered with the specific AMS, as well as the identifiers of the agents that live in each instance. The resource properties of the AgentContainer Service are (a) the name of the AgentContainer and (b) a list of the agent identifiers that currently live in the AgentContainer. Lastly, the DF Service exposes as a single resource property the list of agents that have been registered with it and a description of the services they provide.

The monitoring functions of the framework (such as agent state monitoring) are performed by using the WS-Notification [17] implementation of the Globus Toolkit 4. The AgentContainer Service implements the NotificationProducer portType of the WS-Notification standard, while the AMS Service acts as a notification consumer, being notified about the changes in the AgentContainer resource properties. The DF Service also implements the NotificationProducer portType, so that an agent can be notified about services provided by other agents.

3.2 Agent Creation and Migration

Agent classes can be dynamically loaded by using the dynamic class loading characteristic of the Java language. A class that implements the Agent interface can be dynamically loaded at run time and act as a mobile agent, migrating to remote nodes, or sending and receiving ACL messages.

Agent migration is accomplished by exploiting the Java Object Serialization API. The Agent interface implements the java.io.Serializable interface. An instance of the Agent interface can be a) serialized in a sequence of bytes, b) transferred through the network, c) deserialized at the destination AgentContainer and d) reloaded in local memory. The form of mobility that the agent system supports is characterized as weak mobility, since the execution state of an agent can not be captured and retrieved, and only state data can be migrated along with the agent code [18]. Finally, the migration process is sender-initiated. This means that an agent that needs to migrate has to interact with the AMS Service by calling the "migrate" operation provided by the service.

3.3 Agent Communication

Agent communication is achieved by ACL message exchange. In our agent system, ACL messages are instances of a Java class containing fields that correspond to the abstract parameter message payload identified in [19]. A future plan is to use the XML specification for FIPA ACL message representation [20] and transform the message format into an XML document, which can be sent by encapsulating it in the body of a SOAP message.

The message delivery mechanism is as follows: at first, the sender agent forwards the message to the AgentContainer he resides in. If the receiver agent is currently living in the same AgentContainer, the message is inserted in the local message queue of that agent. Otherwise, the message is forwarded to the AMS Service, which sends the message to the AgentContainer the receiver agent currently resides in. Interacting with the AMS Service for message delivery is a straightforward approach, since agent monitoring information is available at the AMS. Nevertheless, using such a centralized approach for message delivery can result in several bottle-necks. As a future enhancement, an individual messaging service can be inserted in the architecture, which will have to keep track of the location of the agents and send pending messages as appropriate.

4. THE GRID SERVICE MOBILITY FRAMEWORK

The work presented in this paper mainly concentrates on the development of a mobility enabling environment for grid services. As mentioned earlier, a grid constituted entirely of stationary services can to some extent be considered to be rigid and inflexible. By integrating mobility into grid services, a certain level of agility may be achieved. The goal is to provide the grid environment with the mobility capability, while ensuring that the standardization achieved for description, discovery and access to stateful grid services is still valid. In this context, we build on the agent system described in the previous section and propose an architecture for the development of transparently mobile grid services.

In the following paragraphs we present some issues about mobility integration and describe the proposed framework components and architecture, along with a potential use case scenario.

4.1 Mobility Integration Issues

In order for a mobile grid service to be functional and effective, certain issues have to be addressed. The first and foremost issue that needs to be resolved is the actual manner that mobility is achieved. This is roughly equivalent to the problem of process migration during runtime. An important characteristic to the service migration problem is that the application to be moved is deployed in a service container. This means that for a service to migrate to a remote computer node, the service must firstly be

undeployed from the container it currently resides in, its code and state (assuming a weak mobility scheme) must be retrieved and transferred over a network and finally the service must be redeployed to the web service container at the destination node, with its state initialized to the previously retrieved state. Obviously, such an approach is not only technically challenging, but also poses certain problems to service addressing and discovery. This is due to the fact that a moving endpoint is associated with the service instance, which means that special treatment is needed in order for the discovery system to obtain consistent information about the current address of the service and in order for any client to access the service as well.

Alternatively to moving the whole service instance between different containers, it is possible to separate the service interface from the service functionality implementation, and to use mechanisms for the migration of the functionality only. This approach has the advantage that the deployed service instance itself does not need to be migrated, and thus undeployment and redeployment of the service is not necessary; nevertheless, by enabling mobility for the functionality implementation, certain advantages can still be deduced. For example, one may consider the common problem of load imbalance among the nodes of a grid system. By exploiting mobility operations, part of the load can be moved from over-loaded nodes to under-loaded nodes, resulting in more efficient resource utilization. Moreover, standardized ways for interacting with the service may still be used, as the functionality can be exposed as a grid service interface.

The approach chosen to achieve mobility integration is based on the mobile agent paradigm. A software agent or even a more complex agent system can be used to wrap the functionality of the service and move between grid nodes as appropriate with the purpose of completing its execution. One of the main advantages of building on the mobile agent paradigm is that it offers the potentiality to develop flexible mechanisms and complex interactions between application components.

Another important issue that arises from the adoption of the agent approach to service mobility is communication with the service. Communication between agents is asynchronous, while interaction with a web service can be either synchronous or asynchronous. Due to the superposition of the web service layer on top of an asynchronous communication layer, it could be beneficial to invoke mobile service operations in an asynchronous way, for example by using polling schemes. Otherwise, mechanisms for ensuring that web service invocation is blocked until asynchronous operations are completed are obviously demanded.

4.2 Mobile Grid Service Components

The components that constitute a mobile grid service, as well as the interactions and relations between them, are shown in Figure 2. The three main components of a mobile grid service are namely: the Mobile Grid Service Interface, used for the communication with the clients outside the platform, as well as the ServiceGateWayAgent and the ServiceAgentSystem, residing at the core part of the platform, named AgentContainer Service.

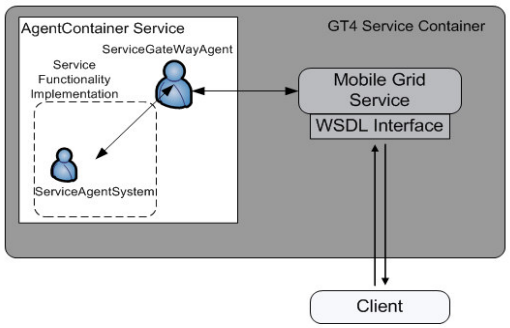

Figure 2. Mobile Grid Service Components

In the following the Mobile Grid Service Components are described in detail:

Mobile Grid Service Interface: The Mobile Grid Service Interface is the component that acts as an access point to the mobile grid service for an external client. It is an ordinary stateful web service that provides a standard WSDL interface describing available operations and exposed resource properties and can be accessed through standard SOAP messages. The difference from a typical web service is that the functionality of the provided operations is not embodied in the service implementation. Instead, each service operation interacts with the ServiceGateWayAgent in order to request from the ServiceAgentSystem to perform certain computations on its behalf. Our purpose is to enable access to the mobile grid service by exploiting all available standardization and that is the main advantage of using a classic grid service as a front end.

ServiceGateWayAgent: The ServiceGateWayAgent is a stationary agent, which resides in the AgentContainer Service that is deployed in the same grid container as the Mobile Grid Service Interface component. It acts as a proxy between the Mobile Grid Service Interface and the ServiceAgentSystem that implements the actual service functionality. The main reason that the Service Interface interacts with the ServiceGateWayAgent component is that it can offer a single point of access to the service functionality that is implemented at the ServiceAgentSystem component. This is due to the fact that the ServiceAgentSystem component may be more complicated than just a single agent and can be consisted of several agents forming complex behaviors. The main disadvantage of this approach is that the ServiceGateWayAgent is potentially a single point of failure. A solution to this problem would be the existence of multiple instances of the ServiceGateWayAgent, though this approach would make the architecture more complex. The necessary interactions with the service functionality implementation can thus be hidden from the service interface component, resulting in a more modular architecture. The ServiceGateWayAgent receives ordinary ACL messages from the Mobile Grid Service Interface requesting a specific operation and accordingly translates and forwards the request to the appropriate agent of the ServiceAgentSystem in the form of an ACL message. The result of the request is returned to ServiceGateWayAgent also as an ACL message and is communicated to the Mobile Grid Service Interface by means of a standard web service client that invokes for example a “set” method.

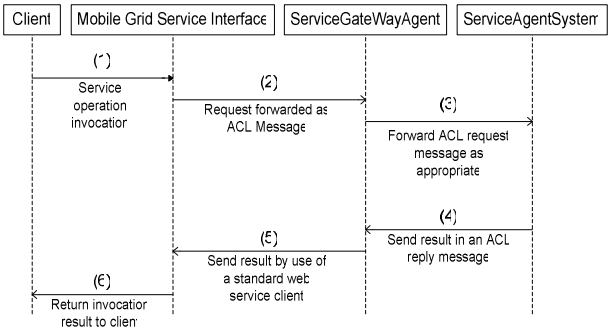

Figure 3. Interaction between framework components

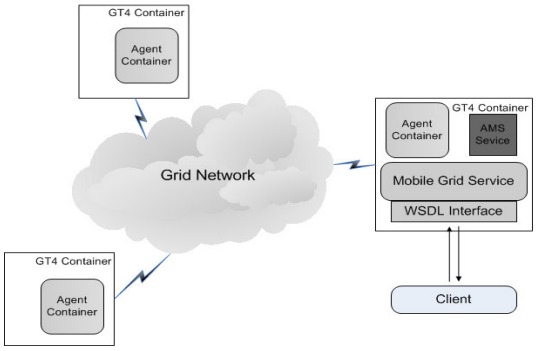

Figure 4. Overall System Architecture

ServiceAgentSystem: The ServiceAgentSystem is the component that actually encapsulates the service functionality. It can be a single agent or a more complex structure composed of several cooperating agents. The operations that can be performed by the ServiceAgentSystem may be triggered by ACL request messages sent by the ServiceGateWayAgent. These messages may indicate for example a method to invoke along with the corresponding arguments. Dynamic method invocation can be achieved by exploiting the Java Reflection API. Furthermore, certain behaviors and appropriate ontologies need to be developed for ServiceGateWayAgent and ServiceAgentSystem to interact efficiently (area for future work).

Figure 3 shows a sequence diagram that demonstrates the abstract interactions between the components of the framework when an operation invocation for the mobile grid service occurs.

This sequence diagram corresponds to the case that the invocation of an operation exposed by the Mobile Grid Service Interface is conducted in a synchronous way. In case of asynchronous invocation, the Mobile Grid Service Interface may return an address for a blackboard service where the result will be posted, along with an identifier for retrieving the result.

4.3 Overall Architecture

The overall architecture of the proposed system is depicted in Figure 4. According to OGSA, a grid system is considered to be composed of a collection of grid services. These services are provided by grid service containers residing in different hosts that are connected with each other over a network. Service migration to remote hosts can be achieved by utilizing the mobility mechanisms provided by the proposed GriSBAP system. In order for a service to be able to move to a different service container, an AgentContainer Service must have been deployed to it, so that agent migration can be supported.

The interface of the mobile service is deployed only to one service container and acts as the access point to any external client that needs to interact with the service functionality. Service functionality is thus exposed by a standard WSDL interface and is attached to a standard endpoint for the whole service lifecycle. On the other hand, the service functionality implementation can be distributed dynamically to all available service containers with an AgentContainer Service instance residing in them.

In Figure 4, the Mobile Grid Service Interface is shown to reside in the service container that the AMS Service is deployed. This is due to the fact that the interface needs to communicate with the ServiceGateWayAgent by sending ACL messages. As described in section 4, communication in the GriSBAP system is handled by the AMS, thus it is more efficient to deploy the Mobile Grid Service Interface on the same service container as the AMS Service in order to minimize communication costs. Similarly, the ServiceGateWayAgent resides in the AgentContainer Service that runs on the same service container as the Mobile Grid Service Interface.

Finally, service discovery for the proposed mobile grid services can be accomplished by using accepted standard procedures. Since the front end of the service is actually a standard GT4 stateful web service, protocols such as UDDI [21] and tools like Globus MDS [22] may be normally used for service discovery and resource monitoring.

4.4 Use Case

Key issues in the development of efficient grid middleware are resource monitoring and load balancing. A number of approaches to these problems have been proposed in the literature [23 - 25] based on the mobile agent paradigm. The proposed systems commonly consist of several agents residing on available nodes, which monitor cpu load or other resources, and certain agents that schedule tasks appropriately, or encapsulate tasks and move to available nodes as needed in order to complete their execution. In the architecture proposed in this paper, such an agent system can be considered to be the functionality of a resource monitoring and task-scheduling service and corresponds to the ServiceAgentSystem described previously. A service interface can expose operations for job submission and retrieval of resource monitoring information, so that a user may access the functionality of the service (for example submit jobs) using standard web service clients.

5. CONCLUSIONS AND FUTURE WORK

In this paper, an attempt towards integrating mobility into the OGSA grid environment by exploiting the mobile agent computing paradigm has been presented. The GriSBAP system, which is an agent platform conforming to FIPA standardization and that consists solely of stateful web services has been introduced. Moreover, several issues concerning mobility integration to grid services have been discussed. Building on

GriSBAP, a framework architecture that enables the functionality of a service to move among different grid containers is proposed. Mobility integration can be accomplished by separating the implementation of a grid service in three different levels: service functionality implementation as a mobile agent system, operation exposure through a standard WSDL interface and interconnection between these two levels through a gateway agent. A potential use case scenario describing the whole life cycle of a resource monitoring and task-scheduling service for grid systems has also been presented and analyzed, emphasizing the advantages of the proposed solution.

In future work, the GriSBAP system will be expanded by implementing more efficient communication mechanisms and by importing further features of FIPA standardization, especially interaction protocols that are essential for synthesizing complex agent behaviors. Furthermore, an API for the creation of mobile grid services as described in this paper will be developed, and other ways of mobility integration will also be investigated.

6. ACKNOWLEDGMENTS

The work presented in this paper has been funded by the project PENED 2003(03ED/226). The project is co financed 75% of public expenditure through EC – European Social Fund, 25% of public expenditure through Ministry of Development - General Secretariat of Research and Technology and through private sector, under measure 8.3 of operational program "COMPETITIVENESS" in the 3rd Community Support Program.

7. REFERENCES

[1] I. Foster, C. Kesselman, S. Tuecke. “The Anatomy of the Grid: Enabling Scalable Virtual Organizations.” International Journal of Supercomputer Applications, vol. 15, No. 3, 2001, pp. 200-222.

[2] I. Foster, C. Kesselman, J. Nick and S. Tuecke,”The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration”, Globus Project, 2002

[3] OASIS Web Services Resource Framework (WSRF) TC, http:// www.oasis-open.org/committees/wsrf/

[4] L. Moreau, “Agents for the Grid: A Comparison with Web Services (Part I: Transport Layer)”, Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on, 21-24 May 2002, pp. 220 – 228

[5] Moreau L., Avila-Rosas A., Dialani V., Miles S. and Liu X., “Agents for the Grid: A Comparison with Web Services (part II: Service Discovery)”, In Proceedings of Workshop on Challenges in Open Agent Systems, Italy, (2002), pp. 52-56.

[6] Zahreddine W., Mahmoud Q.H., “An agent-based approach to composite mobile Web services”, 19th International Conference on Advanced Information Networking and Applications, 2005. AINA 2005, vol. 2, 28-30 March 2005 pp. 189 - 192

[7] Bellavista P., Corradi A., Monti S., “Integrating Web services and mobile agent systems”, 25th IEEE International Conference on Distributed Computing Systems Workshops, 2005, 6-10 June 2005, pp. 283 – 290.

[8] M.O. Shafiq, H.F. Ahmad, H. Suguri and A. Ali, “Autonomous Semantic Grid: Principles of Autonomous Decentralized Systems for Grid Computing”, IEICE Transactions on Information and Systems 2005, E88-D(12), pp. 2640-2650.

[9] Sze-Wing Wong, Kam-Wing Ng, “A middleware framework for secure mobile grid services”, Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, 2006, vol. 2, 16-19 May 2006.

[10] I. Foster, N.R. Jennings, C. Kesselman, “Brain meets Brawn: why grid and agents need each other”, Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004, pp. 8-15.

[11] Foundation for Intelligent Physical Agents, http://www.fipa.org/

[12] I. Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems”, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, 2005, pp. 2-13.

[13] The Globus Alliance, http://www.globus.org/

[14] FIPA Agent Management Specification http://www.fipa.org/specs/fipa00023/

[15] Simple Object Access Protocol (SOAP), http:// www.w3.org/TR/soap/

[16] Web Service Definition Language (WSDL), http:// www.w3.org/TR/wsdl

[17] OASIS Web Services Notification (WSN) TC, http://www.oasis-open.org/committees/wsn/

[18] A. Fuggetta, G.P. Picco, G. Vigna, “Understanding Code Mobility”, IEEE Transactions on Software Engineering, vol. 24, no. 5, May 1998, pp. 342-361.

[19] FIPA ACL Message Structure Specification, http://www.fipa.org/specs/fipa00061/

[20] FIPA ACL Message Representation in XML Specification, http://www.fipa.org/specs/fipa00061/SC00061G.pdf

[21] OASIS Universal Description, Discovery and Integration (UDDI) Specifications, http://www.oasis-open.org/committees/uddi-spec/

[22] Globus Monitoring and Discovery System (MDS), http://www.globus.org/mds/

[23] S.D. Desic, D. Huljenic, “Agents Based Load Balancing with Component Distribution Capability”, 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 21-24 May 2002, pp. 354 - 358.

[24] J. Cao, D.P. Spooner, S.A. Jarvis, S. Saini, G.R. Nudd, “Agent-based grid load balancing using performance-driven task scheduling”, International Parallel and Distributed Processing Symposium, 22-26 April 2003.

[25] Li Chunlin and Li Layuan, “Agent framework to support the computational grid”, Journal of Systems and Software, Volume 70, Issues 1-2, February 2004, Pages 177-187