Service-Based Integration of Grid and Multi-Agent Systems Models

Clement Jonquet, Pascal Dugénie, Stefano A. Cerri

Service-Based Integration of Grid and Multi-Agent Systems Models

Clement Jonquet[1], Pascal Dugenie[2], and Stefano A. Cerri[2]

[1] Stanford Center for Biomedical Informatics Research (BMIR)
Stanford University School of Medicine
Medical School Office Building, Room X-215
251 Campus Drive, Stanford, CA 94305-5479 USA
jonquet@stanford.edu

[2] Laboratory of Informatics, Robotics, and Microelectronics of Montpellier (LIRMM)
National Center of Scientific Research (CNRS) & University Montpellier 2
161 Rue Ada, 34392 Montpellier, France
{dugenie,cerri}@lirmm.fr

Abstract. This position paper addresses the question of integrating GRID and MAS (Multi-Agent Systems) models by means of a service oriented approach. Service Oriented Computing (SOC) tries to address many challenges in the world of computing with services. The concept of service is clearly at the intersection of GRID and MAS and their integration allows to address one of these key challenges: the implementation of dynamically generated services based on conversations. In our approach, services are exchanged (i.e., provided and used) by *agents* through *GRID* mechanisms and infrastructure. Integration goes beyond the simple interoperation of applications and standards, it has to be intrinsic to the underpinning model. We introduce here an (quite unique) integration model for GRID and MAS. This model is formalized and represented by a graphical description language called Agent-Grid Integration Language (AGIL). This integration is based on two main ideas: (i) the representation of agent capabilities as Grid services in service containers; (ii) the assimilation of the service instantiation mechanism (from GRID) with the creation of a new conversation context (from MAS). The integrated model may be seen as a formalization of agent interaction for service exchange.

1 Introduction

The GRID and MAS communities believe in the potential of GRID and MAS to enhance each other because these models have developed significant complementarities [1]. One of the crucial explorations concerns the substitution by an agent-oriented kernel of the current object-oriented kernel of services available in Service Oriented Architectures (SOAs), including GRID. The Service Oriented Computing (SOC) community agrees that such a change will really leverage SOC scenarios by providing new types of services [2]. This key concept of *service* is clearly at the intersection of the GRID and MAS domains and thus may motivate an integration.[3] GRID is said to be the first

[3] [1] foresees services as the core 'unifying concept' that underlies GRID and MAS (also historically suggested by [3] and [4]).

distributed architecture (and infrastructure) really developed in a service-oriented perspective: Grid services are compliant Web services, based on the dynamic allocation of virtualized resources to an instantiated service [5]. Actually, GRID acquired major importance in SOA by augmenting the basic notion of Web Service with two significant features: service state and service lifetime management. Whereas Web services have instances that are stateless and persistent, Grid service instances can be either stateful or stateless, and can be either transient or persistent.[4] On the other hand, agents are said to be autonomous, intelligent and interactive entities who may use and provide services (in the sense of particular problem-solving capabilities). Actually, agents have many interesting characteristics for service exchange: they are reactive, efficient, adaptive, they know about themselves, they have a memory and a persistent state, they are able to have conversation, work collaboratively, negotiate, learn and reason to evolve, deal with semantics associated to concepts by processing ontologies, etc. MAS and SOC communities recently turned to one another considering the important abilities of agents for providing and using dynamic composed services, semantic services, business processes, etc. (see [7] for a recent overview of SOC challenges). Web services are often criticized because they are no more than Remote Procedure Calls (RPC) which have no user adaptation, no memory, no lifetime management, no conversation handling capabilities (simple request/answer interaction). They are passive, they lack semantics and they do not take into account the autonomy of components. The SOC community has realized that the notion of service has to surpass HyperText Transfer Protocol, current SOA standards (Web Service Definition Language (WSDL), Simple Object Access Protocol (SOAP), Universal Description Discovery and Integration (UDDI)), RPCs and eXtensible Markup Language (XML) to be enriched by results from other research domains such as information systems, concurrent systems, knowledge engineering, interaction and, especially, GRID and MAS.

To provide a service means to identify and offer a solution (among many possible ones) to the problem of another. The next generation of services will consist of dynamically generated services, i.e., services constructed on the fly by the service provider according to the conversation it has with the service user. In *Dynamic Service Generation* (DSG), term suggested by [8, 9], the user (human or artificial) is not assumed to know exactly what the provider (also human or artificial) can offer him. He finds out and constructs step by step what he wants based on the service provider's reactions. The central idea of DSG is that a service may be based on a conversation. Actually, DSG highlights the idea of processing something new instead of merely delivering something that already exists. In everyday life, when somebody needs new clothes, *buying ready-to-wear clothes* is analogous to asking for a product, whereas *having clothes made by a tailor* is analogous to requiring a service to be generated. Singh and Huhns [7] talk about *service engagement*, instead of simple method invocation. In [8, 9] we present the STROBE model as an agent representation and communication model designed and constructed in order to develop dynamically generated services. The shift from the currently limited perspective in service exchange scenarios to DSG is the topic addressed by this paper. It introduces a service based GRID-MAS integrated model to help to

[4] Grid service specifications are described both by Open Grid Service Architecture (OGSA) [5] and Web Service Resource Framework (WSRF) [6].

go towards this DSG vision by providing a common integration to help the community designing service architectures that benefits from both MAS and GRID interesting service features. In order to summarize our thoughts at the intersection of the three domains (GRID, MAS and SOC), we identify two key ideas:

- GRID and MAS have each developed a service oriented behaviour, therefore the concept of service may represent a common integration;
- New needs in service exchange scenarios are clearly highlighted and may be met by integrating GRID and MAS complementarities.

In [9, 10], we introduce the *Agent-Grid Integration Language* (AGIL) as a GRID-MAS integrated systems description language which rigorously formalizes both key GRID and MAS concepts, their relations and the rules of their integration with graphical representations and a set-theory formalization. AGIL is both an integration model and a description language (i.e., a sort of UML for GRID-MAS integrated systems). In this position paper we present quickly the main ideas and principles of AGIL integration model.

2 Brief state of the art

2.1 Brief GRID overview

The GRID aims to enable *flexible, secure, coordinated resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organization*. Actually, it was originally designed to be an environment with a large number of networked computer systems where computing (Grid computing) and storage (data Grid) resources could be shared as needed and on demand. GRID provides the protocols, services and software development kits needed to enable flexible, controlled resource sharing on a large scale. This sharing is, necessarily, highly controlled, with resource providers and users defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A GRID system is naturally highly dynamic and should be able to adapt at runtime to changes in system state as resource availability may fluctuate. Grid users are members of *virtual organizations/communities*. A virtual organization (VO) is a dynamic collection of individuals, institutions and resources sharing common goals, bundled together in order to share resources and services.

GRID technologies have evolved from ad hoc solutions, and de facto standards based on the Globus Toolkit, to Open Grid Services Architecture (OGSA) [5] which adopts Web service standards and extends services to all kind of resources (not only computing and storage). Foster et al. call service: *a (potentially transient) stateful service instance supporting reliable and secure invocation (when required), lifetime management, notification, policy management, credential management, and virtualization*. OGSA introduces two major aspects in SOA by distinguishing service factory from service instance. In other words, services are instantiated with their own dedicated resources and for a certain amount of time. These characteristics enable (i) service state management: Grid services can be either stateful or stateless; (ii) service lifetime management: Grid services can be either transient or persistent. More recently, the Web

Service Resource Framework (WSRF) [6] defines uniform mechanisms for defining, inspecting, and managing stateful resources in Web/Grid services. WSRF models Grid service as an association, called a WS-Resource, between two entities: a stateless Web service, which does not have state, and stateful resources which do have state. A stateful service has an internal state that persists over multiple interactions.

2.2 Integration related work

Some work has already been proposed for using agents to enhance Web services or integrating MAS & SOC. For a detailed comparison between these two concepts see, for example, [11]. [12] points out some drawbacks of Web services which significantly distinguish them from agents. According to us, different kind of approaches may be distinguished in agent-Web service integration:

Distinct view of agents and Web services. Agents are able both to describe their services as Web services and to search/use Web services by using mappings between MAS standards and SOA standards [11, 13–15]. This approach is often based on a gateway or wrapper which transforms one standard into another. As the main approach in agent standardization is the one of Foundation for Intelligent Physical Agents (FIPA), this work only considers FIPA agents and resolves relationships between SOA and FIPA standards. A particularly difficult factor in this approach is communication. The challenge consists of bridging the gap between semantically reach asynchronous based agent communications and semantically poor synchronous based Web service communications.

Uniform view of agents and Web services. Agents and Web services are the same entities. All services are Web services and they are all provided by agents (i.e., the underpinning program application is an agent-based system) [16, 17].

MAS to support SOC/SOA mechanisms. This approach is not directly interested in agent service-Web service interaction but rather in the use of MAS to enhance SOAs. For example, [18] discusses the use of agents for Web services selection according to the quality of matching criteria and ratings.

MAS-based Business Process Management. Workflow or service orchestration is analogous to interaction protocol in agent communication. Both terms describe a common interaction structure that specifies a set of intermediate states in the communication process as well as the transitions between these states. The applicability of MAS to workflow enactment has been noted by [19]. More specifically, [14] makes a strict comparison between workflow (with Business Process Execution Language for Web Services) and interaction protocol (as FIPA has defined them). Conversation or service choreography is also analogous to agent conversation. Conversations are long-lived high-level interactions which need a peer-to-peer, proactive, dynamic and loosely coupled mode of interaction. Using agent conversations to enhance service exchange is an active research topic [20–22].

There is an increasing amount of research activity in GRID and MAS convergence taking place.[5] The use of agents for GRID was very early suggested by [3]. The authors

[5] See, for example, 'Agent-Based Cluster and Grid Computing' workshops, 'Smart Grid Technologies' AAMAS workshops, the Multi-Agent and Grid System journal.

specifically detail how agents can provide a useful abstraction at the Computational Grid layer and enhance resource and service discovery, negotiation, registries, etc. MAS has also been established in 2001 as a key element of the Semantic Grid [4]. And more recently, why GRID and MAS need each other has been established by [1]. The authors emphasizes the overlap in problems that GRID and MAS address but without sharing research progress in either area: *an integrated Grid/agent approach will only be achieved via a more fine-grain intertwining of the two technologies*. Using MAS principles to improve core GRID performances (e.g., directory services, scheduling, brokering services, task allocation, dynamic resource allocation and load balancing) is a very active topic in the MAS community, for example: (i) MAS-based GRID for resource management [23–26]; (ii) MAS-based GRID for VO management [27].

However, none of this work proposes a real integration of MAS and GRID. Rather, they focus on how MAS and AI techniques may enhance core GRID functionalities. Our vision of a GRID-MAS integration is not a simple interoperation of the technologies. It goes beyond a simple use of one technology to enhance the other. We aim to adopt a common approach for the integration to be able to benefit from the most relevant aspects of both GRID and MAS. This common approach is centred on the concept of service.

3 Agent-Grid Integration Language

3.1 AGIL's concepts

This section defines progressively each AGIL's concepts coming from both GRID and MAS and integrated together in a common and relevant manner. Notice that key GRID concepts presented in this section have been established according to the OGSA or WSRF specifications. Similarly, key MAS concepts have been established by different approaches in the MAS literature [22, 28] but especially the STROBE model [8, 9]. As we are focussing on concepts, we adopt the most convenient terminology from these sets of specifications. AGIL's integration model is graphically presented in Figure 2 and explained in the following paragraphs:

In SOC, a *service* is an interface of a functionality (or capability) compliant with SOA standards. Figure 1 presents services we aim to formalize and their associated symbols. Stateless services are quite restrictive: they are synchronous (i.e., messages can not be buffered and do block the sender or receiver), point-to-point (i.e., used by only one user) and interact via simple one-shot interaction (i.e., request/answer). A stateless service does not establish a conversation. Instead, it returns a result from an invocation, much like a function. Stateful services required additional consideration: they are instantiated with a given set of resources. They can be persistent or transient (instantiated for a given period of time, this period may change dynamically). Transient services are instantiated by a service factory whereas persistent services are created by out-of-band mechanisms such as the initialization of a new service container. Stateful services may be multipoint (i.e., used by several users) and may interact by simple one-shot interaction or long-lived conversation. Stateful services may be synchronous or asynchronous.

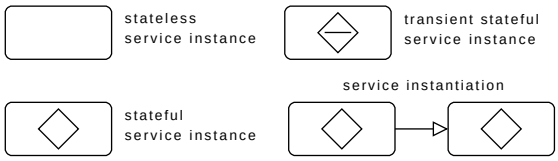

Fig. 1. Representation of key service concepts

GRID is a resource-sharing system. Grid resources are contributed by *hosts*. A host is either a *single host* (i.e., a direct association between a *computing resource* and a *storage resource*) or a *coupled host* (i.e., an aggregation of different single hosts and/or coupled hots). The sharing of these resources is implemented by the *virtualization* and *reification*[6] of these resources in a *service container*. A (Grid) service is an interface of a functionality (or capability) compliant with SOA standards. An *service instance* is included in a hosting environment in order to exist and to evolve with their own private contexts (i.e., set of resources). This is the role of the service container which is the reification of a portion of the virtualized resource available in a secure and reliable manner. A service container contains several types of services. A service may instantiate another service in the same or different service container. Each service is identified by a *handle*. Since a container is a particular kind of service, it is created either through the use of a service factory or by the direct core GRID functionality. A service container is allocated to (and created for) one and only one group of *agents*, called a *Virtual Organization* (VO). Each agent may be a *member* of several VOs. The relation between a VO and a service container is embodied by an *authorization service* which formalizes the VO-dedicated policies of service by members. The authorization service may be viewed as a MxS matrix, where M corresponds to the number of members of the VO, S to the number of currently active services, and the matrix nodes are deontic rules. These rules permit the accurate specification of the right levels for a member on a service (e.g., permissions, interdictions, restrictions etc.).[7] In order to participate in GRID, hosts and agents must hold a *X509 certificate* signed by a special authority.

An *agent* possesses both intelligent and functional abilities. These are represented respectively by the agent *brain* and *body*. The brain is composed of a set of rules and algorithms (e.g., machine learning) that give to the agent learning and reasoning skills. It also contains the agent knowledge, objectives, and mental states (e.g., Belief-Desire-Intention). The body is composed of a set of *capabilities* which correspond to the agent's capacity or ability to do something, i.e., to perform some task. These capabilities may be interfaced as Grid services in the service container that belongs to a VO an agent is a member of. In the agent's body, these capabilities may be executed in a particular conversation context called a *cognitive environment*. A cognitive environment contains several capacities. An agent may have several cognitive environments

[6] Resource virtualization and reification is done at the core GRID level (middleware). The rest of GRID core level mechanisms (e.g., container, authorization, etc.) are themselves described by a single unit: the Grid service.

[7] Such authorization service may be for instance, a Community Authorization Service (CAS) or Virtual Organization Membership Service (VOMS).

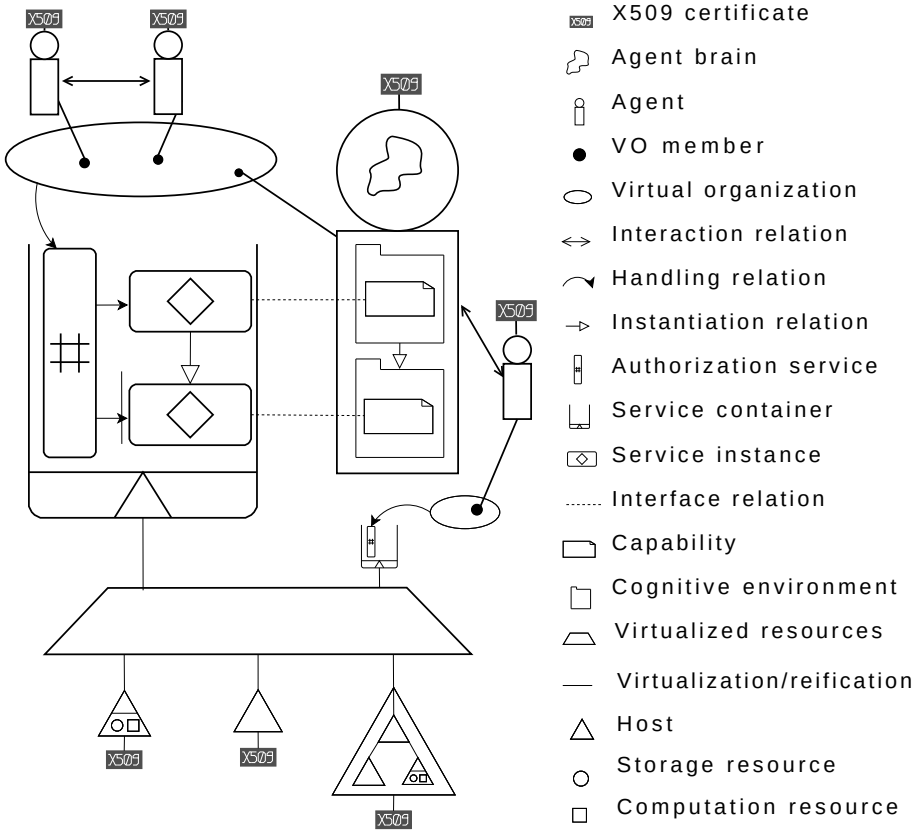

Fig. 2. AGIL integration model and graphical representation

which correspond to the different conversation contexts and languages it develops by *interaction* with other agents. Service exchange interaction are defined when an agent uses the service another agent provides.

In MAS, when an agent has a conversation, it dedicates a part of its state to this conversation. It is called the conversation context [8, 22]. For example, during service exchanges, the service provider maintain a set of explicit interaction contexts, corresponding to each of its users. Conversations and their states are represented in the STROBE model by cognitive environments. We can explore further the concept of cognitive environments,[8] which is a relatively new, but very important, concept related to the STROBE agent and communication model [8, 9, 29, 30]. In the STROBE model agents are able to interpret communication messages in a given conversation context that includes an interpreter, dedicated to the current conversation. We show how communication enables dynamic changes in these dedicated contexts and how these inter-

[8] The term environment is used here in its programming-language meaning, that is to say, a structure that binds variables and values. It does not mean the world surrounding an agent.

preters can dynamically adapt their way of interpreting messages (meta-level learning by communication). Each time an agent receives a message, it selects the unique corresponding cognitive environment dedicated to the message sender in order to interpret the message. When an agent receives a message for the first time, it *instantiates* a new dedicated conversation context for this agent by creating a new one or sharing an already existing one. This instantiation mechanism is similar than the one existing in Grid services.

Having dedicated contexts and thus dedicated capabilities, is demonstrated a good means to go towards DSG. In other agent architectures, a cognitive environment may simply be viewed as a conversation context. The same concept of putting the communication contexts at the centre of the agent architecture in which it interprets messages appears also in [22], which assumes that each agent in service exchanges may separately maintain its own internal context of the conversation state.

3.2 Integration of GRID and MAS concepts

The integration of key GRID and MAS concepts concerns five major aspects:

1. The term *agent* is used to uniformly denote Artificial Agent, Human Agent and Grid user. They are active entities involved in service exchanges. They are considered autonomous, intelligent and interactive. In particular, by viewing Grid users as agents, we may consider them as a potential artificial entities.

2. The term *VO* unifies the concept of VO in GRID and the concept of group in MAS. This is a dynamic social group (virtual or not). It is the context of service exchanges.

3. The two concepts of *service* and *capability* are linked together with a new one-to-one relation between them called the *interface relation* (represented by a dotted line in Figure 2). A Grid service is seen as the interface of a capability published in a service container and with allocated resources. An agent has a set of capabilities it may transform into Grid services available in the different VOs it is a member of. The process of 'transforming' or 'publishing' a capability into a service is called the *servicization process*.[9] When a capability is servicized, it means:

- the interfacing of this capability with SOA standards i.e., mainly WSDL/SOAP;
- the addition (possibly by using an add-service service) of this service to the VO's service container by assigning it a handle and by allocating it private resources;
- the requesting of the VO's authorization service to add an entry for this service (the agent has to decide the users' right levels);
- the publishing of the service description in the VO's registry, if it exists;
- the notification to the VO's members of the VO that a new service is available;
- etc., according to VO or service container local rules.

When an agent servicizes one of its capability into a service available for a VO it uses a set of services of this VO. Each of the previous step of the servicization process

[9] We can say that as GRID virtualizes resources and reifies them in a service container, an agent virtualizes capabilities and reifies them in a service container.

is achieved using a specific VO local service (e.g., interfacing, adding, notifications services). This servicization process is not discrete but continuous. Service and capability keep aligned in time one another. For example, if the capability of the agent changes, then the service changes at the same time. With this viewpoint, an agent can provide different VOs with different services. Notice also that a service is agent-specific, that means that only one agent executes (i.e., provides) the service in a container. However, it does not prevent another agent of the VO from providing the same type of service. What is important in this servicization process is that it abstracts on the kind of agent involved. Both AAs and HAs transform their capabilities in the VO's service container modulo different (graphical) interfaces. For example, an AA may servicize its capability to compute square roots (i.e., a function that receives an integer as a parameter and returns a float as result), and a HA may servicize its pattern-recognition capability (i.e., a function that receives an image as a parameter and returns a concept as result). Notice that the service and the capability lifetimes are not necessarily the same. Even if a service is transient in a service container the corresponding capability maybe persistent in the agent's body.

Remark – Grid resources are available for services (i.e., servicized capabilities) execution. The agent itself is still executed autonomously with its own resources and process (e.g., on an agent platform such as JADE).

4. The key GRID idea of service instantiation is integrated with the MAS idea of creating a dedicated conversation context. The processes are the same but viewed differently. The new conversation context contains the new capability and the service provider applies the servicization process on it in order to make available the new service instance for the service user(s). The association between the conversation context (stateful) and the including capability (stateless) is view as a WS-Resource.[10] Integrating these two instantiation mechanisms make capabilities to benefit from standardization, interoperation and allocated resources from GRID, and Grid services to benefit from a dedicated context of execution and local conversation representation from MAS.

5. Agent-agent interactions include all other kinds of interactions (Grid user-Grid service, Grid service-Grid service, agent-agent, etc.). These interactions are realized by means of asynchronous message passing between agents. There is two kind of interactions:

Direct agent-agent interaction. Messages are exchanged directly from agent to agent. These are interactions in a general sense, i.e., any interaction, standardized or ad hoc, protocol guided or not, semantically described or not, long-lived or one-shot, etc. These interactions may occur within a VO, but also outside it;

Through-service agent-agent interaction. They occur during service exchange. Messages are exchanged from agent to agent through a service. These are interactions that an agent may have with another agent, without directly communicating with the other agent but instead via the service interface this second agent offers in the VO's service container. These 'through-service interactions' occur only within a VO.

[10] In order to map exactly the STROBE mechanisms to OGSA and WSRF mechanisms, we should say that a new cognitive environment may be viewed as a new WS-Resource, i.e., a dedicated association between capabilities and stateful resources.

What is important in this integrated model is to consider how a service may be adapted by a service provider agent for a service user agent, in order to implement DSG. We identify four ways:

1. The service provider agent adapts the dedicated service according to its interactions with service user agent;
2. The service provider agent may offer another service to change or adapt the original service (meta-level);
3. The service provider agent may use dynamic intelligent reflection rules to change the service it is currently providing;
4. Direct agent-agent interactions may occur between the service user agent and the service provider agent and within these interactions (1) and (3) may occur in a pure ad hoc form (not via service).

3.3 Discussions and benefits for GRID, MAS and SOC.

Some AGIL advantages may be summarized:

• There is no real standard in the MAS community to describe agents' capabilities between different agents or MAS. The integration will help MAS developers in presenting and interfacing agents' capabilities, and therefore augment MAS interoperation and standardization.

• This integrated model does not restrict MAS or GRID in any way. In particular, it does not prevent direct agent-agent interactions and thus, for example, it does not prevent agents to perform tasks to one another in a purely ad hoc manner. This is important if we want the integration to be followed by numbers of MAS approaches and models; these models can keep their internal formalisms for their internal operations.

• In this integration, VO management benefits from both GRID and MAS organizational structure formalisms, e.g., Agent-Group-Role [28], CAS service, X509 certificate, etc.

• Service exchanges in this integrated model benefit from the important agent communications abilities, e.g., dealing with semantics, ability to hold a conversation, etc. The challenge in MAS of modelling conversation not by a fixed structure (interaction protocol) but by a dynamic dialogue becomes the same that dynamically composing and choreographing services in business processes as suggested by DSG.

• This integrated model subsumes a significant number of the MAS-based GRID approaches cited in section 2.2. Indeed, thanks to the reflexivity of GRID, which defines some GRID core functionalities as (meta-)Grid services (e.g., service container, auhtorization service), we may consider these core GRID services as executed also by agents. This establishes an important part of the MAS-based GRID approaches which use MAS techniques to enhance core GRID functionalities.

4 Conclusion

Identifying key factors to demonstrate the convergence of MAS and GRID models is not an easy task. We point out that the current state of GRID and MAS research activities is sufficiently mature to enable justifying the exploration of the path towards an

integration of the two domains. At the core of this integration is the concept of service. The bottom-up vision of service in GRID combined with the top-down vision of service in MAS bring forth a richer concept of service, integrating both GRID and MAS properties. We put this enhanced concept of service into the perspective of Dynamic Service Generation (DSG).

In our integrated model, we consider agents exchanging services through VOs they are members of: both the service user and the service provider are considered to be agents. They may decide to make available one of their capabilities in a certain VO but not in another. The VO's service container is then used as a service publication/retrieval platform (the semantics may also be situated there). A service is executed by an agent with resources allocated by the service container. We sum-up here AGIL's two main underlying ideas:

- The representation of agent capabilities as Grid services in service containers, i.e., viewing Grid service as an 'allocated interface' of an agent capability by substituting the object-oriented kernel of Web/Grid services with and agent oriented one;
- The assimilation of the service instantiation mechanism – fundamental in GRID as it allows Grid services to be stateful and dynamic – with the dedicated cognitive environment instantiation mechanism – fundamental in STROBE as it allows one agent to dedicate to another one a conversation context.

In [31] we propose Agora, an architecture model that uses GRID to deploy collaborative ubiquituous spaces for collective intelligence. AGIL integration model is demonstrated as a key element for such an infrastructure. AGIL model is feasible considering today's state of SOC, MAS and GRID technologies. Integrating these aspects according to the guidelines given in this paper seems to us a good way to capitalize past, present and future work in order to simplify the scenarios and use fruitfully the power of distributed services, exchanged among communities of humans and artificial agents.

References

1. Foster, I., Jennings, N.R., Kesselman, C.: Brain meets brawn: why Grid and agents need each other. In: 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'04. Volume 1., New York, NY, USA (July 2004) 8–15
2. Huhns, M.N., Singh, M.P., Burstein, M., Decker, K., Durfee, E., Finin, T., Gasser, L., Goradia, H., Jennings, N.R., Lakkaraju, K., Nakashima, H., Parunak, V., Rosenschein, J.S., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T., Zavala, L.: Research directions for service-oriented multiagent systems. Internet Computing **9**(6) (November-December 2005) 65–70
3. Rana, O.F., Moreau, L.: Issues in building agent based computational Grids. In: 3rd Workshop of the UK Special Interest Group on Multi-Agent Systems, UKMAS'00, Oxford, UK (December 2000) 1–11
4. Roure, D.D., Jennings, N.R., Shadbolt, N.: Research agenda for the Semantic Grid: a future e-science infrastructure. Technical report, University of Southampton, UK (June 2001) Report commissioned for EPSRC/DTI Core e-Science Programme.
5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the Grid: an Open Grid Services Architecture for distributed systems integration. In: Open Grid Service Infrastructure WG, Global Grid Forum, The Globus Alliance (June 2002)

6. Foster, I., Frey, J., Graham, S., Tuecke, S., Czajkowski, K., Ferguson, D.F., Leymann, F., Nally, M., Sedukhin, I., Snelling, D., Storey, T., Vambenepe, W., Weerawarana, S.: Modeling stateful resources with Web services. Whitepaper Ver. 1.1, The Globus Alliance (May 2004)
7. Singh, M.P., Huhns, M.N.: Service-Oriented Computing, Semantics, Processes, Agents. John Wiley & Sons (2005)
8. Jonquet, C., Cerri, S.A.: The STROBE model: Dynamic Service Generation on the Grid. Applied Artificial Intelligence, Special issue on Learning Grid Services **19**(9-10) (October-November 2005) 967–1013
9. Jonquet, C.: Dynamic Service Generation: Agent interactions for service exchange on the Grid. PhD thesis, University Montpellier 2, Montpellier, France (November 2006)
10. Jonquet, C., Dugenie, P., Cerri, S.A.: Agent-Grid Integration Language. Multiagent and Grid Systems (2008) Accepted for publication - In press expected Number 1 Volume 4, 2008.
11. Moreau, L.: Agents for the Grid: a comparison with Web services (part 1: the transport layer). In Bal, H.E., Lohr, K.P., Reinefeld, A., eds.: 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID'02, Berlin, Germany, IEEE Computer Society (May 2002) 220–228
12. Huhns, M.N.: Agents as Web services. Internet Computing **6**(4) (July-August 2002) 93–95
13. Lyell, M., Rosen, L., Casagni-Simkins, M., Norris, D.: On software agents and Web services: usage and design concepts and issues. In: 1st International Workshop on Web Services and Agent Based Engineering, WSABE'03, Melbourne, Australia (July 2003)
14. Buhler, P.A., Vidal, J.M.: Integrating agent services into BPEL4WS defined workflows. In: 4th International Workshop on Web-Oriented Software Technologies, IWWOST'04, Munich, Germany (July 2004)
15. Greenwood, D., Calisti, M.: Engineering Web service - agent integration. In: IEEE Systems, Cybernetics and Man Conference, SMC'04. Volume 2., The Hague, Netherlands, IEEE Computer Society (October 2004) 1918–1925
16. Ishikawa, F., Yoshioka, N., Tahara, Y.: Toward synthesis of Web services and mobile agents. In: 2nd International Workshop on Web Services and Agent Based Engineering, WSABE'04, New York, NY, USA (July 2004) 48–55
17. Peters, J.: Integration of mobile agents and Web services. In: 1st European Young Researchers Workshop on Service-Oriented Computing, YR-SOC'05, Leicester, UK, Software Technology Research Laboratory, De Montfort University (April 2005) 53–58
18. Maximilien, E.M., Singh, M.P.: Agent-based architecture for autonomic Web service selection. In: 1st International Workshop on Web Services and Agent Based Engineering, WSABE'03, Sydney, Australia (July 2003)
19. Singh, M.P., Huhns, M.N.: Multiagent systems for workflow. Intelligent Systems in Accounting, Finance and Management **8**(2) (June 1999) 105–117
20. Maamar, Z., Mostéfaoui, S.K., Lahkim, M.: Web services composition using software agents and conversations. In Benslimane, D., ed.: Les services Web. Volume 10 of RSTI-ISI. Lavoisier (2005) 49–66
21. Hanson, J.E., Nandi, P., Levine, D.W.: Conversation-enabled Web services for agents and e-business. In: 3rd International Conference on Internet Computing, IC'02, Las Vegas, NV, USA (June 2002) 791–796
22. Ardissono, L., Goy, A., Petrone, G.: Enabling conversations with Web services. In: 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'03, Melbourne, Australia, ACM Press (July 2003) 819–826
23. Manola, F., Thompson, C.: Characterizing the agent Grid. Technical report 990623, Object Services and Consulting, Inc. (June 1999)
24. Cao, J., Jarvis, S.A., Saini, S., Kerbyson, D.J., Nudd, G.R.: ARMS: an Agent-based Resource Management System for Grid computing. Scientific Programming, Special issue on Grid Computing **10**(2) (2002) 135–148

25. Shen, W., Li, Y., Ghenniwa, H.H., Wang, C.: Adaptive negotiation for agent-based Grid computing. In: 1st International Agentcities Workshop on Challenges in Open Agent Environments, Bologna, Italy (July 2002) 32–36
26. Manvi, S.S., Birje, M.N., Prasad, B.: An agent-based resource allocation model for computational Grids. Multiagent and Grid Systems **1**(1) (2005) 17–27
27. Patel, J., Teacy, W.T.L., Jennings, N.R., Luck, M., Chalmers, S., Oren, N., Norman, T.J., Preece, A., Gray, P.M.D., Shercliff, G., Stockreisser, P.J., Shao, J., Gray, W.A., Fiddian, N.J., Thompson, S.: Agent-based virtual organisations for the Grid. Multiagent and Grid Systems **1**(4) (2005) 237–249
28. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. In Giorgini, P., Müller, J.P., Odell, J., eds.: 4th International Workshop on Agent-Oriented Software Engineering, AOSE'03. Volume 2935 of Lecture Notes in Computer Science., Melbourne, Australia, Springer-Verlag (July 2003) 214–230
29. Cerri, S.A.: Cognitive Environments in the STROBE model. In Brna, P., Paiva, A., Self, J., eds.: European Conference in Artificial Intelligence and Education, EuroAIED'96, Lisbon, Portugal (October 1996) 254–260
30. Cerri, S.A.: Shifting the focus from control to communication: the STReam OBjects Environments model of communicating agents. In Padget, J., ed.: Collaboration between Human and Artificial Societies, Coordination and Agent-Based Distributed Computing. Volume 1624 of Lecture Note in Artificial Intelligence. Springer-Verlag, Berlin, Germany (1999) 74–101
31. Dugénie, P.: UCS, Ubiquitous Collaborative Spaces on an infrastructure of distributed resources. PhD thesis, University Montpellier 2, Montpellier, France (December 2007)