Movement and Interaction in Semantic GRIDs: Dynamic Service Generation for Agents in the MIC* Deployment Environment

Abdelkader Gouaich, Stefano A. Cerri

Movement and interaction in semantic GRIDs: dynamic service generation for Agents in the MIC* deployment environment

GOUAICH Abdelkader and CERRI Stefano A.
LIRMM, CNRS and University Montpellier II
161, Rue Ada, 34392 Montpellier, France
http://www.lirmm.fr
{*gouaich,cerri*}*@lirmm.fr*

Abstract

We present in this position paper the foundations of the MIC* model and deployment environment as they have emerged in the last years, relate them to current evolutions on semantic GRID dynamic service generation, as reflected by OGSA/I and, more recently, by WSRF, and propose an integrated view with the previously proposed STROBE model for communicating agents leading to a quite simple yet very promising architecture that may include Human agents in the loop, uninspectable as most artificial agents, yet behaving autonomously by interacting. Modeling Human agents seems to us of high importance in the high level service generation required by complex, semantically rich applications, such as those of e-Learning, e-Science or e-Commerce envisaged in the years to come.

1. INTRODUCTION

The distributed computing field has evolved profoundly as a consequence of the globalisation and democratisation of the network infrastructures. Nowadays, distributed software systems are no more developed only for a single organisation, but are potentially accessible world-widely from any point of the Internet. Besides, by the appearance of mobile networks, the domain of software systems has been enlarged from offices and personal houses to any geographical point that may access the wireless network. This change on the technological environment of distributed software systems implies also changes on their models, infrastructures and software engineering processes. For instance, [28] lists some of the properties that should be considered in order to build distributed software system in the outlined current context (including our own comments in parenthesis)

- situatedness: The locality of communication and interaction implies that the software system global functionality depends on the local context of its elementary components (therefore the global system has to adapt itself to the local interactions).
- openness: The software system has no more a defined clear barrier and static structure; it is permanently evolving by merging or rejecting sub-systems (therefore the global system's functionalities have to be capable to evolve with respect to the local evolutions).
- heterogeneity: In a large connected world several actors and organisations with different goals build software components. Hence, several heterogeneous entities have to collaborate or coordinate their actions to achieve their own goals (therefore interoperability is not just a wish but a prerequisite for any sensible design).
- autonomy of the components: Autonomy of a software components is an important feature that should be remembered when designing large open distributed software systems. Hence, software components have to be considered as black boxes, and lack of knowledge on their internals structure make them behaving autonomously since their reaction for an external stimulus is not fully predictable [1]. The debate on autonomy concerns often just software components, eg: objects vs agents: when the human is in the loop, however, as it is necessarily the case for current distributed systems, the autonomy of humans is not anymore an issue under debate, but a fact to be considered

true. Human agents participate to the system's evolution but their behaviour is not accessible but from their communicative processes.

Perhaps the most important feature of current and future distributed software systems - in particular within the Semantic GRID context - is their foreseen ability to integrate and coordinate on the fly the available services. In fact, the open service's composition is becoming a central concern addressed by different communities such as: Semantic Web services, Multi-Agent Systems and Grid-computing. The idea is to consider services offered on the Internet as virtual organisations and businesses engaged in long lasting transactions (interactions and movement). As it is the case for human organisations and businesses, these virtual organisations offer purposely their resources and know-how to build more complex and rich services with an added value that could not have been offered by the entities separately. This organisational model, based on the division and specialisation of labour, has already proven its success for human organisations and businesses by achieving ambitious scientific and industrial goals related to complex problem solutions. Still, the prerequisite for coordination among open and autonomous services is to have a flexible and reliable infrastructure that defines their interactions, guarantees their autonomy; and protects them from harmful actions that challenge the commonly established and admitted norms. The purpose of this paper is

- to present the current models and infrastructures for the deployment of large distributed systems;
- to present our formal model named MIC* and its prototypical software implementation: a deployment environment for distributed system . This model is intended to capture the main features of the Internet and mobile context such as the autonomy of the services and the on the fly composition of sub-systems;
- to relate the MIC*: Movement Interaction Calculus model (and its deployment environment) to current concerns of GRID's dynamic service generation, in particular related to mobility and interaction.

2. MODELS AND INFRASTRUCTURES FOR OPEN (AND LARGE) DISTRIBUTED COMPUTING

This section reviews briefly some of the current models and infrastructures for the development of open and large distributed software systems that open opportunities for the deployment of advanced electronic services.

2.1. The Object Request Broker Architecture (CORBA)

The Object Request Broker Architecture (CORBA) is a key component of the OMG's Object Management Architecture (OMA). The OMA is composed of an Object Model and a Reference Model. The object model defines how distributed software components can be described, while the reference model identifies interaction among these components. The role of the Object Request Broker (ORB) in the OMA model is facilitating communication between objects. Before CORBA 2.0, commercial ORB products did not interoperate since the OMG specifications did not mandate any particular data formats or protocols for inter-ORB communications. Since, CORBA 2.0 ORB interoperability became a focus of the OMG and the developed GIOP architecture and its implementation for the Internet world (IIOP) allows several CORBA sub-systems to be interconnected world-widely. However, the software components interact only by method calls and until CORBA 3.0 only three ways of method invocation were considered: synchronous invocation; one-way invocation; and deferred synchronous invocation. The OMG has recognised the limits of these mechanisms and have introduced a higher-level communication model such as asynchronous messaging services in CORBA 3.0. This communication model is more suitable for large-scale and possibly mobile distributed systems where the communication media are not reliable.

2.2. Formal models

Several formal frameworks have been proposed to model distributed systems. Among these models one can cite the followings: The chemical abstract machine (CHAM) [5]; The Pi calculus [25]; and Ambient [7]. The CHAM is introduced by Berry and Boudoul and considers a distributed system as a chemical solution evolving by reactions according to global transformation laws. CHAM addresses only the specification of distributed systems without any ambition for implementation. Indeed, the entities are passive and fully

controlled by static and global evolutions' laws. Milner introduced the Pi calculus in the eighties. The Pi calculus models processes that communicate by message exchanges through named channels. Ambient is the most recent model introduced by Luca Cardelli in 1998. An ambient is defined as a location where calculation happens. Ambients are mobile entities and can nest other Ambients and processes. Besides the Ambient model, the author describes his decomposition of the context of distributed systems as three mental images: the LAN vision, representing an administrable and perfectly controlled area; the WAN vision, representing an area where the global supervision is impossible; and the Mobile vision, representing the physical and logical movement of the components between several areas. Obviously, the properties of these three mental images are totally different and one has to study carefully the context of its application in order to make the most appropriate design choices.

2.3. GRID computing

GRID technologies support the sharing and coordinated use of interconnected resources in dynamic virtual organisations. In the beginning, the focus was on computing power: the dynamic creation from geographically and organisationally distributed components of a virtual computing system able to generate 'autonomously' the service consisting of dynamically allocated computing resources for heavy processes [16]. In the beginning GRID technologies were designed for advanced science and engineering oriented services consisting mainly of processing power. Since then [15] the ambition of GRID computing has shifted its original goal to address any kind of electronic services such as those related to interactive processes, as it is the case, for instance, of persistent transactions with distributed Information Systems, e-Commerce and e-Learning applications, Ambient intelligence and so on. The main logical components of a GRID architecture are the following: computing elements, representing the computers and clusters that run users' processes and jobs; storage elements, that represent the storage space where temporary and persistent data are stored; user interfaces, represent the services that permit to the final user to access and use the offered resources of the GRID architecture; resource brokers, these are the central elements of a GRID architecture that handle the users' requests, allocate the desired resources and track the execution of the users' requests and finally inform the users about the result of their requests; information services, offering a 'yellow page' service that inform about the available resources and location within the GRID. The current central debate on GRID technologies (GLOBUS OGSA/I [2, 3] and WSRF [4]) concerns the most suitable architecture that hosts two contradictory properties of distributed systems: their independence on the asynchronicity of message exchange requiring a purely functional behaviour in order to avoid a heavy synchronization control or the risk of unforeseeable failures and the need for modeling, deploying and using persistent, stateful transactions (interactions, conversations). This debate will be further discussed in the subsequent sections, as the MIC* model seems to represent a progress for a grounded solution for these concerns.

2.4. Multi-Agent Systems

The multi-agent system (MAS) paradigm considers a distributed software system as a virtual organisation of autonomous and interacting entities named agents. The concept of 'autonomy' of computational entities with respect to stimuli (messages) received from the external environment is probably the most important feature that this paradigm has introduced in the field of distributed systems. In fact, the agents are no more considered as objects that have to answer any external message by method invocation: they may have their own internals laws and goals that can prevent them to do so. So, an external observer [1] have always to consider different situations that are consequence of the autonomy of its interlocutor. The FIPA [14] has developed standards to build agent-based applications and to make them interoperable. The FIPA specifies the logical components of an agent platform that are the agent management system (AMS), the directory facilitator (DF), the agent communication channel (ACC); and a standardised semantic interaction language: the FIPA-ACL. The content of the messages that are exchanged among agents is specified using ontologies and may be expressed in different content languages. Within the MAS paradigm, the two major properties of modern distributed systems have been put in the foreground - even if certainly not yet fully developed -: autonomy and interaction. At the moment, one of the most important discussion concerning the architecture of future MASs regards the two opposite views where to put the 'intelligence' required for autonomy and processing of interactions: in the Agents themselves, or an individually centered architecture (BDI theories

[1] This observer is for instance the user of the service offered by this agent.

[24]) versus a socially centered architecture, where the core of the 'intelligence' is within the socially shared 'meaning' emerging from individual commitments caused by message exchanges [21, 27, 9]

2.5. Mobile code

The mobile code field studies software systems where the software components change locality during their life cycle [26, 6]. According to [8] three majors paradigms are used to build mobile code systems: remote evaluation, code on demand, and mobile agents. Within the remote evaluation paradigm, the know-how, or instructions, are locally available but the necessary resources for code executions are missing. Therefore, the know-how is remotely executed on an execution environment containing these resources. The code on demand is interpreted as the availability of the necessary resources, but the lack of the know-how. Consequently, the software component downloads the know-how and executes it locally. Lastly, within the mobile agent paradigm, the know-how is the exclusive property of an agent. When an agent misses specific resources in order to complete its tasks, it moves as a whole in order to reach an execution environment containing them.

2.6. Tuple spaces

The tuple space paradigm has been introduced by researchers at Yale University where Linda [18] –the first tuple space-based system– has been conceived. A tuple space-based system is composed by the following components: tuples - a tuple is basically a list of typed fields - ; fields, that are *actual* when they hold a value or *formal* when no value is contained; tuple spaces - a tuple space is an abstract storage location where tuples are deposited and retrieved by processes - ; processes - that represent the active entities that store and retrieve tuples from tuple spaces - . The tuple space paradigm appeared as an interesting alternative to the RPC-like paradigm. In fact, the interaction is decoupled in space and asynchronous in time. This property is suitable for building distributed software systems where the communication media are unreliable [17]. However, Linda-like systems have known some scalability problems when the number of processes and tuples is important. In fact, since the tuple space does not own any particular structure [2] it is regarded as a flat data set that may be costly in access time to retrieve a particular tuple.

2.7. Web services

Web services offer platform-independent techniques for: describing software components to accessed through an XML-based service description language (WSDL), methods for accessing these components that are independent from the transport layer (SOAP), and discovery methods that enable the identification of relevant service providers in a service registry (UDDI). Besides, other efforts have been made recently in order to compose web services through XML-based standards such as Web Services Flow Language (WSFL). Web services is a promising approach that has facilitated the integration of various heterogeneous components, but it is still influenced by the RPC-like and CORBA approaches and other considerations were not addressed such as the autonomy of the components and the offer of a semantic interaction language among the components. Recently, an evolution both of OGSA/I and Web services has been proposed within the Web Service Resource Framework WSRF, a joint initiative from the Globus Alliance, IBM, Fujitsu and Hewlett Packard [4].

3. THE MIC* MODEL

3.1. A brief description of MIC*

By considering the presented properties of the current context of distributed systems and the autonomy feature of the components, we have developed an algebraic model of an *deployment environment*. A deployment environment is defined as the container of the autonomous and active entities that are called *agents*. Our original choice concerning the architecture has been to develop a model of an environment that holds autonomous agents *without considering their internals architectures*. The other interesting feature of our model, that has motivated the algebraic approach, is the ability to compose formally several sub-deployment environments to other deployment environments. So, the system's composition is explicitly and formally described in the model itself. The consequences of both innovative choices will be discussed

[2] The tuple space may also be considered as the shared memory in the blackboard architectures

and motivated in section 4 where we relate our MIC* to current advances in semantic GRID models and technologies.

3.2. Intuitive concepts of MIC*

As said previously within the context of open systems, agents have to be considered as completely closed to internals inspection. Consequently, the study of agent deployment environments considers the fundamental features of agents such as autonomy and interaction not from the viewpoint of the 'internal architecture' of agents - that is not accessible -, but from the 'social' traces as those available and inspectable by the environment (messages, or better: interaction objects, hereafter defined). The control of interaction among agents by the environment is independent from their internals architecture.

In order to be exchanged, information is usually encoded using explicit interaction objects, which carries the information from a place to another. These interaction objects own a particular structure. The first (mathematical) abstraction is to define an empty interaction object that carries no information. Besides, the interaction objects can be composed formally and commutatively to represent a group of interaction objects occurring simultaneously. Consequently, interaction objects have naturally a structure of a commutative monoid $(\mathcal{O}, +)$ with a commutative composition law $+$ and a neutral element 0. The interaction rules among the agents are defined contextually. This introduces the concept of the *interaction space*. The interaction space defines an abstract location where interaction among interaction objects holds. So, agents are perceived in their deployment environment only through their interaction objects. An agent may be present in several interaction spaces: this defines its coordinates in the deployment environment. When an agent is not present in a certain interaction space, its representation is equal to the empty interaction object 0. When this value changes, this is perceived as a 'movement' of the agent inside the interaction space. Therefore, the mobility of an agent can be defined as the movement of its interaction objects among interaction spaces. In order to easily model this notion of mobility, it is necessary to define negative interaction objects, as will be hereafter shown. As agent's location is denoted by the presence in the interaction space of some agent's interaction object, whenever an agent moves outside an interaction space it is natural to impose that its representation is reduced to 0, i.e: the empty interaction object. This can be expressed by $x - (x) = 0$. So, the interaction object's structure is no more a simple commutative monoid but a commutative group $(\mathcal{O}, +)$. To summarise, intuitively the concepts of a deployment environment are interaction objects defined as a commutative group $(\mathcal{O}, +)$ encoding information exchanged among agents through interaction spaces. Interaction spaces, represented by $\mathcal{S}$, are active entities that define the interaction laws among interaction objects. These concepts are captured in an algebraic structure, $\mathcal{T}$, composed: by two matrices called the inbox matrix and the outbox matrix, and a memory vector.

The rows of the outbox and inbox matrices represent the agents, the columns represent the interaction spaces, and the elements of these matrices are the interaction objects that represent respectively the inboxes and the outboxes of each agent in each interaction space. The rows of the memory vector represent the agents themselves, the elements are the memories of each agent. We assume that the memory vector exists, but we cannot inspect it. No further assumptions are made on the memory vector.

3.3. MIC^* dynamics

A MIC* element $T \in \mathcal{T}$ is a instantaneous snapshot of the agent deployment environment state. The dynamical aspects of the deployment environment are represented by the set of applications defined on: $e : \mathcal{T} \to \mathcal{T}$ that link any elements $T \in \mathcal{T}$ to the next deployment environment state $T \mapsto e(T) \in \mathcal{T}$. Among applications defined on $\mathcal{T} \to \mathcal{T}$, we have selected some particular classes that have a special semantics:

Interaction (φ): From an external point of view, two agents are considered as interacting when the perceptions of an agent - its inbox - are influenced by the emissions of another one - its outbox -. Consequently, interaction evolutions should modify only the perceptions of agents according to the emissions of other agents within a defined interaction space. The set of all interaction evolutions is represented as φ.

Movement (μ): The mobility of an agent is defined as the mobility of its interaction objects among different interaction spaces. During a movement no interaction object is created or lost. In fact, this is an interesting feature to prevent incoherent duplications by guaranteeing that an entity actually disappears from its original location and appears in its destination. Therefore, a movement switches only the places of the agent outboxes between interaction spaces and leave them globally invariant. The set of all movement evolutions is represented by μ.

Computation (γ): Computation is an agent's internal, non inspectable process. The deployment environment is assumed to not have access to the internals structure of agents. The only way to observe that an agent has conducted a computation is when it changes autonomously its outboxes within interaction spaces. The set of all computation evolutions is represented by γ.

The dynamics of the deployment environment is viewed as the successive application of these evolution laws from a starting element considered as the initial state.

3.4. Composing deployment environments in MIC*

Thanks to the algebraic modelling, formal laws are defined to model composition, that is junctions and disjunctions of systems, of several MIC* deployment environments. When the sub-systems are joined, agents that are in the same interaction spaces may interact in order to achieve their goals, while when the deployment environment is split into different sub-environments, agents that do not belong to the same deployment environment may not interact. Let us suppose that two separated environments [3] $T_1 \in \mathcal{T}/\mathcal{L}$ and $T_2 \in \mathcal{T}/\mathcal{L}$ are defined as follows[4]:

$$\begin{aligned} T_1 &= \left(\begin{bmatrix}[o_1]_a\\ [0]_b\end{bmatrix}_s \quad \begin{bmatrix}[0]_a\\ [o_2]_b\end{bmatrix}_t\right)\left(\begin{bmatrix}[i_1]_a\\ [0]_b\end{bmatrix}_s \quad \begin{bmatrix}[0]_a\\ [i_2]_b\end{bmatrix}_t\right) \\ T_2 &= \left(\begin{bmatrix}[o_3]_c\end{bmatrix}_s \quad \begin{bmatrix}[0]_c\end{bmatrix}_t \quad \begin{bmatrix}[o_4]_c\end{bmatrix}_v\right)\left(\begin{bmatrix}[i_3]_c\end{bmatrix}_s \quad \begin{bmatrix}[0]_c\end{bmatrix}_t \quad \begin{bmatrix}[i_4]_c\end{bmatrix}_v\right) \end{aligned}$$

T_1 contains two interaction spaces s and t and two agents a and b. T_2 contains three interaction spaces s, t, v and only one agent c. By using the linear notations T_1 and T_2 are written as follows:

$$\begin{aligned} T_1 &= \begin{cases} \text{outbox:} & o_1.a@s + o_2.b@t \\ \text{inbox:} & i_1.a@s + i_2.b@t \end{cases} \\ T_2 &= \begin{cases} \text{outbox:} & o_3.c@s + o_4.c@v \\ \text{inbox:} & i_3.c@s + i_4.c@v \end{cases} \end{aligned}$$

Now let us suppose that these two terms are composed. This is translated in the MIC* model as the composition under the $+$ law:

$$\begin{aligned} T_{\{T_1,T_2\}} &= T_1 + T_2 \\ &= \begin{cases} \text{outbox:} & o_1.a@s + o_2.b@t + o_3.c@s + o_4.c@v \\ \text{inbox:} & i_1.a@s + i_2.b@t + i_3.c@s + i_4.c@v \end{cases} \\ &= \begin{cases} \text{outbox:} & (o_1.a + o_3.c)@s + o_2.b@t + o_4.c@v \\ \text{inbox:} & (i_1.a + o_3.c)@s + i_2.b@t + i_4.c@v \end{cases} \\ &= \left(\begin{bmatrix}[o_1]_a\\ [0]_b\\ [o_3]_c\end{bmatrix}_s \quad \begin{bmatrix}[0]_a\\ [o_2]_b\\ [0]_c\end{bmatrix}_t \quad \begin{bmatrix}[0]_a\\ [0]_b\\ [o_4]_c\end{bmatrix}_v\right)\left(\begin{bmatrix}[i_1]_a\\ [0]_b\\ [i_3]_c\end{bmatrix}_s \quad \begin{bmatrix}[0]_a\\ [i_2]_b\\ [0]_c\end{bmatrix}_t \quad \begin{bmatrix}[0]_a\\ [0]_b\\ [i_4]_c\end{bmatrix}_v\right) \end{aligned}$$

The composed deployment environment $T_{\{T_1,T_2\}}$ is made of all interaction spaces and agents found in sub-environments. One can notice that when agents are in the same interaction space they are still located in the same interaction space in the composed environment. So, they can interact in this new environment using interaction mechanisms defined in §3.3 and more in details in [20, 19].

[3] $\mathcal{T}/\mathcal{L}$ represents MIC* structure where the memory vector is ignored

[4] The reader has to assume the algebraic properties of the given notations such as the commutitivity of elements and thier associativity. The expression $x.p@s$ is interpreted as the agent p has x as an inbox or outbox (depending on the line) in the interaction space s.

3.5. Implementation of the MIC* model

The MIC* model and its principles have been implemented using C/C++ and are available at *http://www.sourceforge.net/projects/mic*. Several applications have been implemented especially for ubiquitous and mobile scenario where a movement simulator simulates the movement of human users in a 3D space. Users are supposed to carry computers, where software systems are running, equipped with short-range communication facilities. When two users are geographically side-by-side the deployment environments are composed on the fly and the deployed software agents may interact in order to achieve their goals and offer their services. When the users move away, the transported deployment environments are disjoined. This scenario is supposed to capture the most important difficulties and constraints that one may find to develop large scale open and distributed software services on the Internet.

4. FOUNDATIONS FOR AN OPEN GRID HUMAN SERVICE ARCHITECTURE: OGHSA

The fundamental purpose of this paper, as stated in the introduction, is to evaluate the interest of the MIC* model and its principles for the progress of semantic GRIDs. This endeavour is central within the LEGE-WG project, but even more within the ELEGI EU 6th FP Integrated Project, both focussing on advances in semantic GRID with respect to the complex set of potential e-Learning applications. In this section we will present our vision on the synergies between MIC* , semantic GRID, agents and e-Learning as it becomes more and more clear by working in LEGE-WG and ELEGI. The dynamic generation of services by agents on the GRID was previously introduced and discussed in a few introductory papers: [12, 22, 10, 13].

Recently, Globus Alliance, HP and Fujitsu have announced a proposal: Web Service Resource Framework WSRF (*http://www.globus.org/wsrf*) that one has to take into account seriously in order to relate future models and architectures with both needs and alternatives already investigated by the community. The fundamental lessons learned from the WSRF proposal consists, for us, of two major needs for successful service generation:

1. the required stateful and persistent nature of services. This is clearly coupled with the interactive (conversational) nature of services that cannot be denied in real applications. Bank transactions, a subset of transactions with distributed Information Systems, but also e-commerce, e-learning etc.
2. the required flexible compositionality of services. It is hard or perhaps impossible to compose dynamically state dependent components, much easier to compose state independent ones, i.e: functions with no side effects. Looking at WSRF, we notice (page 8 of: Modeling Stateful Resources with Web Services) a brief classification of service models with respect to memory and state into 'truly stateless services', 'conversational services' and finally 'stateless services that act upon stateful resources'. WSRF explicitily adresses the third model, while the second one is better considered by other approaches, as context/headers, WS-Coordination and WS-Policy. Hereafter some considerations on each of the three.
    - 'truly stateless services': These are represented as pure functions. The advantage of easy composition of purely functional services comes at the cost that they can hardly represent state, unless we introduce in the model the concept of delayed evaluation and stream [11] as it is the case in the STROBE model for communicating agents.
    - 'conversational services': These are the most generic stateful services. Hard to be realized within a distributed and asynchronous context, heavy to be supported and maintained, they however maintain their fundamental interest for the most advanced applications. We believe that higher level services such as those emerging from semantically rich domains will require this model to co-exist with the other ones. At the moment, the only viable and generic solution for deploying services that conserve state seems for us the use of Continuations [23] within developments -such as those around the STROBE model - that adopt first class continuations in order to model easily context switching among conversational threads [13].
    - 'stateless services that act upon stateful resources': These are services of an intermediate complexity - in terms of needs for state and memory - as seems to us to be the case in most of the examples offered by [4]. These services are indeed modeled in WSRF by two sets of separate, yet interconnected and communicating entities: Web services, that do not have state, and WS Resources that do have state.

Interactions between Web Services and WS Resources occur through messages adopting a standard syntax called Implied Resource Pattern. We see in the WSRF proposal several trends that join our own proposed models (MIC* and STROBE):

1. Purely functional computational elements are separated from stateful elements, as it is the case for Objects in the STROBE model, where the first-classness nature of Environments 'refactories' the description of computational patterns from their interpretation contexts (the Environments).
2. Computational elements are strictly separated from messages, as it is the case in both STROBE and MIC*. Insofar a set of rules for interaction within interaction spaces can be defined in MIC* for generating the effects of interaction objects within the space in terms of new interaction objects to be consumed by agents, the access to the internal of agents is not necessary. This models potentially a wide spectrum of network services to be offered to unaccessible agents, certainly proprietary artificial agents, but particularily human agents. If an architecture and an infrastructure including human agents has to be developed within ELEGI, as well as in general for the future sociology of the GRID, that architecture will not be able to inspect inside agents, if not in a conversational way and with a very limited spectrum of interactive acts (cf, in WSRF: the acts getReservation, addFlightSegment, removeFlightSegment).
3. These acts may initially consist of those presented in the examples of WSRF: read a state variable, write a state variable. We therefore look at WS Resources as if they were a kind of agents, consisting of a quite limited dispatching capability, but controlling each an Environment in the STROBE sense where variables may be bound to any first class structure available in the model (simple and complex data, but also functions, continuations, other environments, and interpreters: [13].
4. We therefore come to the conclusion that a Web service may be represented by an open structured datum available as an interaction object in one or more interaction spaces in MIC*, possessing no state, but binding itself with stateful WS Resources that are agents managing (reading from and writing into) STROBE Environments where access is only possible through inspectable messages deposited on MIC* .

5. CONCLUSION

In this paper we have tried to present the MIC* model's and deployment environment's foundations, and to relate it to other available models of distributed computation enabling interaction and mobility on networks. Our approach has been to tackle key issues in current GRID and Web Service technologies, such as state and persistency of interactions (or conversations). We have noticed, with a yet quite superficial analysis, an astonishing convergence of MIC* choices and previously developed models for agent's interactive behavious (STROBE) on the one side, and the recent concerns around OGSA/OGSI and WSRF. We therefore conclude that certainly the road is quite long, but there is a quite interesting acceleration of consensus around approaches and concretely adopted research and development directions.

In particular, we have the ambition to have identified at least one basic requirement for including the Human in the loop of future GRID developments, i.e: the necessity of considering human agents as weakly inspectable by interaction, as it is the case for proprietary software agents in a heterogenous system. The major side effect of these considerations, ie: how human interfaces have to be conceived in order to accommodate such a view of Humans in the dynamic generation of services on GRIDs, will be the next conceptual and practical issue to be discussed and exemplified within the ELEGI project as well as the Semantic GRID community.

REFERENCES

[1] GOUAICH Abdelkader. Requirements for achieving software agents autonomy and defining their responsibility. In *Autonomy Workshop at AAMAS 2003*, Melbourne, Australia, 2003.
[2] The Globus Alliance. The open grid services architecture (ogsa). http://www.globus.org/ogsa.
[3] The Globus Alliance. The open grid services infrastructure (ogsi). http://www-unix.globus.org/toolkit.
[4] The Globus Alliance. The ws-resource framework. http://www-fp.globus.org/wsrf.
[5] Gerard Berry and Gerard Boudol. The chemical abstract machine. *Theoretical computer science*, 1992.

[6] Luca Cardelli. A language with distributed scope. In *Conference Record of POPL '95: 22nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages,*, volume 8, pages 286–297, New York, NY, 1995.

[7] Luca Cardelli. Abstractions for mobile computation. *Secure Internet Programming*, pages 51–94, 1999.

[8] Antonio Carzaniga, Gian Pietro Picco, and Giovanni Vigna. Designing distributed applications with a mobile code paradigm. In *Proceedings of the 19th International Conference on Software Engineering*, Boston, MA, 1997.

[9] C. Castelfranchi. Social power a point missed in multi-agent, dai and hci. In *Decentralised AI*, pages 49–62. Y. Demazeau and J. P. Muller, eds. amsterdam edition.

[10] S. A. Cerri. Open learning service scenarios on grids. In *3rd International LeGE-WG Workshop: GRID Infrastructure to Support Future Technology Enhanced Learning*, Berlin, Germany, December 2003. Electronic Workshops in Computing (eWiC).

[11] Stefano A. Cerri. Shifting the focus from control to communication: the STReams OBjects environments model of communicating agents. In *Collaboration between Human and Artificial Societies*, pages 74–101, 1999.

[12] Stefano A. Cerri. Human and artificial agent's conversations on the grid. In *1st LEGE-WG International Workshop on Educational Models for GRID Based Services*, Lausanne, Switzerland., September 2001. Electronic Workshops in Computing (eWiC).

[13] Stefano A. Cerri, Marc Eisenstadt, and Clement Jonquet. Dynamic Learning Agents and Enhanced Presence on the Grid. In *3rd International LeGE-WG Workshop: GRID Infrastructure to Support Future Technology Enhanced Learning*, Berlin, Germany, December 2003. Electronic Workshops in Computing (eWiC).

[14] FIPA. Foundation of intelligent and physical agents, 1996. http://www.fipa.org.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. *Open Grid Service Infrastructure WG*, 2002.

[16] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 15(3), 2001.

[17] Adrian Friday, Stephan Wade, Nigel Davies, and Gardon Blair. The tuple space: An old solution to a new problem?, 1997.

[18] David Gelernter. Generative communication in linda. *ACM Transaction od Programming Languages and Systems*, 7(1):80–112, 1985.

[19] Abdelkader Gouaich and Yves Guiraud. Mic* algebraic structure. Technical report, LIRMM, 2003. http://www.lirmm.fr/~gouaich/research.html.

[20] Abdelkader GOUAICH, Yves GUIRAUD, and Fabien MICHEL. Mic*: An agent formal environment. 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003), 7 2003. Orlando, USA.

[21] Frank Guerin and Jeremy Pitt. Denotational semantics for agent communication language. In *Proceedings of the fifth international conference on Autonomous agents*, pages 497–504. ACM Press, 2001.

[22] S. A. Mahe J. Sallantin P. Lemoisson, S. A. Cerri. Constructive interactions. In *3rd International LeGE-WG Workshop: GRID Infrastructure to Support Future Technology Enhanced Learning*, Berlin, Germany, December 2003. Electronic Workshops in Computing (eWiC).

[23] Christian Queinnec. Inverting back the inversion of control or, continuations versus page-centric programming. *SIGPLAN Not.*, 38(2):57–64, 2003.

[24] A. S. Rao and M. P. Georgeff. (bdi)-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[25] Milner Robin, Parrow Joachim, and Walker David. A calculus for mobile processes, parts 1 and 2. *Information and Computation*, 100(1), 1992.

[26] G.-C. Roman, G. P. Picco, and A. L. Murphy. Software engineering for mobility: A roadmap. *The Future of Software Engineerin,*, pages 241–258, 2000.

[27] Feng Wan and Munindar P. Singh. Commitments and causality for multiagent design. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 749–756. ACM Press, 2003.

[28] Franco Zambonelli and H. Van Dyke Parunak. Signs of a revolution in computer science and software engineering. In *Agent Oriented Software Engineering Workshop at AAMAS 2002*, Bologna, 2002.