SAMBA – An Agent architecture for Ambient Intelligence Elements Interoperability

Arne-Jørgen Berre[1], Giovanna Di Marzo Serugendo[2], Djamel Khadraoui[3], François Charoy[4], George Athanasopoulos[5], Michael Pantazoglou[5], Jean-Henry Morin[6], Pavlos Moraitis[7], Nikolaos Spanoudakis[8]

[1] SINTEF, Forskningsveien 1, Blindern, 0314 OSLO, Norway
arne.j.berre@sintef.no
[2] Birkbeck College,University of London, UK
dimarzo@dcs.bbk.ac.uk
[3] Centre de Recherche Henri Tudor, Luxembourg
djamel.khadraoui@tudor.lu
[4] University Henri Poincaré, France
charoy@loria.fr
[5] National and Kapodistrian University of Athens, Greece
gathanas@di.uoa.gr, michaelp@di.uoa.gr
[6] Korea University Business School, Seoul, Korea
jhmorin@korea.ac.kr
[7] René Descartes University, Paris, France
pavlos@math-info.univ-paris5.fr
[8] Singular Logic SA, Greece
nspan@singularlogic.eu

Abstract. The SAMBA (Systems for AMBient intelligence enabled by Agents) architecture reported here is a conceptual *service-oriented architecture* supporting the *interaction and interoperability* of systems, applications and actors by the notion of an "Ambient Intelligence Element Society". The objective is to provide an *ecosystem infrastructure* supporting the interaction and interoperability of various elements by encapsulating and representing them through *agents* acting as members of an Ambient Intelligence Elements Society, and by using *executable models* at run-time in support of interoperability.

1 Introduction

The vision for Ambient intelligence (AmI) envisages that devices (e.g. nodes, routers, PDAs) and software agents running on those devices, organise themselves

for the benefit of their respective users. Ambient Intelligence builds on three recent key technologies: Ubiquitous Computing, Ubiquitous Communication and Intelligent User Interfaces. Ubiquitous Computing refers to integration of microprocessors into everyday objects like furniture, clothing, white goods, toys, even paint. Ubiquitous Communication enables these objects to communicate with each other and with the user by means of ad-hoc and wireless networking. An Intelligent User Interface enables the inhabitants of the Ambient Intelligence environment to control and interact with the environment in a natural (voice, gestures) and personalised way (preferences, context).

In this paper, we will refer to those inhabitants (i.e. devices and software agents) as *AmI elements* (AmIE). Such AmI elements interoperate and share knowledge or experiences, they gather information (e.g., about road traffic), and they automatically pay amounts of money from e-purses, customise room lighting and temperature, request for references, or build user profiles in loosely coupled environments. Other AmI elements at the same time serve managerial purposes dealing with corporate data in B2B or B2A interactions. These applications are supported by unobtrusive and invisible technology, which is able to take decisions and initiatives, make proposals to the user, and negotiate with others. In addition, in order to support human beings fully without overloading them with requests and information, the underlying technology needs advanced means to let the AmI elements interact and coordinate their work as autonomously as possible. This can only be achieved if AmI elements can interoperate with each other from business point of view as well as from technical and semantic point of view.

More specifically, *business interoperability* concerns issues related to inter-organisational business processes. *Technical interoperability* refers at the knowledge level (e.g. different formats, schemas, and ontology), and at the level of the underlying information and communication technologies (ICT) and systems. *Semantic interoperability* is crucial for interactions among AmI elements that do not know each other, and that have no common design time compliance. Semantic interoperability avoids pre-established (at design-time) agreements on information formats or communication schemas, allowing the discovery and understanding on-the-fly of relevant information needed for allowing interactions to take place. Furthermore, the AmI vision combined with the vision of future Networked Businesses and eGovernment –the ability of doing “Plug-and-Play” of services by 2010 - has a common need for addressing the problem of semantic interoperability.

In this paper, we present a framework called SAMBA (Systems for AMBient intelligence enabled by Agents) that provides a solution to the aforementioned interoperability issues. Section 2 describes the AmIE Society requirements addressed by SAMBA, while Sections 3, 4 and 5 describe respectively the SAMBA Architecture and SAMBA Interoperability Infrastructures. Section 6 highlights the use of the Model Driven approach in the context of SAMBA. Section 7 discusses related works, while Section 8 provides conclusions and a description of future work.

2 AmIE Society Requirements addressed by SAMBA

The AmIE society contains a set of interacting and interoperating AmI elements, represented by *agents*. The SAMBA infrastructure will support the agents' reasoning mechanism, agents' individual strategies and dialogues and will provide a run-time service-oriented infrastructure supporting the execution and social semantic interactions of AmI elements. This infrastructure integrates in a modular way the notions of services, autonomous agents, mobile agents and mobile devices by addressing the following main interoperability requirements between AmI elements.

Semantic interactions. AmI elements participating in ambient intelligence application, or simply working on behalf of some user, need to interact with each other and with their environment in order to realise their individual or collective goals. This interaction implies an understanding of peers' behaviour, functionality, and offered services or requested tasks. These interactions also happen with unknown peers, or with peers with no pre-established agreement at design time.

Autonomous behaviour. Elements situated in an ambient intelligence environment interact autonomously with each other. They are each dedicated to a particular user, but need to interact with other elements in order to realise their functionality. Designing and developing autonomous embedded systems requires careful management of autonomy issues.

Social interactions. Elements participating in an ambient intelligence environment constitute an ecosystem, i.e. a society of autonomous AmI elements that perform activities similar to human behaviour: collaborations, competitions, transactions, negotiations, and contracting. In addition, for security purposes, such elements may gather and share knowledge, information and experience amongst themselves, building a trust-based system. This kind of social behaviour requires that AmI elements are equipped with extra functionalities that go beyond their basic goals. They need to exchange semantic (meaningful) information of different types: functionality, non-functional aspects, quality of service, current state, recommendations, and negotiations.

Discovery and Composition of services. Independently of the underlying actual technology, the interaction model for AmI elements is based on a service-oriented architecture. AmI elements both offer services to and request services from other elements. Such services, which may have a heterogeneous format, require an infrastructure for dynamic advertisement, discovery, retrieval, access and combination to obtain additional functionality irrespectively of the underlying protocols and standards as well as platforms used for their provision and usage.

Security. Security issues are intimately linked to the AmI elements' autonomy and the fact that they reside in a shared environment over which none of the elements have complete control, and of which each element has only a local perception. Nevertheless, AmI elements have to ensure their survival, and reach their goal. Security issues relate to data integrity, confidentiality, authentication, non-repudiation, access control and resources management, and also include trust, risk issues and digital rights and policy management (persistent protection).

Electronic legal transactions. For many domains, not limited just to e-Business, AmI elements may need to engage in legal transactions for accessing, validating, or

authenticating services provided by other elements, or by the environment. These transactions cover aspects related to electronic contracting, rights and policy management, transfer of electronic currency, and validation of such transactions.

SAMBA provides an integrated computing infrastructure enabling semantic interoperability and social interaction among autonomous AmI elements. This infrastructure addresses the above requirements in an intertwined way (mutual understanding, knowledge sharing for handling interoperability, security support, resource management, and electronic transactions).

3 SAMBA Architecture

Context. The SAMBA architecture is currently being discussed in the context of the domain of "Architecture & Platform interoperability" in the INTEROP Network of Excellence [3] project. The vision is to support the interaction and interoperability of various systems, applications and actors by encapsulating and representing them through agents acting as AmI elements in an AmI society. In the Agent group of the aforementioned domain we have worked on how to provide support for semantic interoperability and trust, in the context of Ubiquitous Communication, for personal, business and governmental interactions.

Approach. The key point in the SAMBA approach resides in the separation of concern between the *code* of an AmI element, and the description of its *functional* behaviour, its *non-functional* constraints and requirements (e.g. for low-power devices, reputation, contracting) and *run-time policies*. This description (expressed under the form of a formal specification, active model, or ontology) is available and used at run-time for publishing AmIE capabilities under the form of *services*, for AmIE to request services, and for describing non-functional requirements and policies.

The ecosystem of AmI elements (AmIE society) is supported by the SAMBA architecture depicted in Figure 1. Under the generic term of AmI elements, the SAMBA architecture intends to support the following different kinds of autonomous software: *intelligent agents, mobile agents, mobile devices*, and *Web services*. All the services used are specified in models before they become AmI elements of the AmIE Society having their intrinsic discovery and interaction capabilities based on context, semantic and security interoperability. The SAMBA architecture is composed of 2 interoperability layers bound together by the notion of models used at run-time:

- **AmIE Semantic Interoperability Infrastructure**. This upper layer deals exclusively with models and provides *high-level interoperability* among AmI elements by dealing with contextual, semantic and non-functional aspects. It thus provides AmI elements with capabilities to discover each other, to interact with each other (either directly or not) and to enforce policies.
- **AmIE Technical Interoperability Infrastructure**. This layer is the *execution environment* of the different AmI software. It is responsible for supporting the models by providing the link between the code and the

corresponding model. This layer provides *technical interoperability* among different technologies involved in the AmIE society, essentially through the agents' abstraction paradigm.

The key idea for linking the two infrastructure layers and for clearly handling the different aspects of separation of concerns resides in the use of formal descriptions or models of the different concerns at run-time. These models describe: functional behaviour of AmI components, non-functional execution constraints (particularly for mobile devices), service requests from AmI elements, services provided by AmI elements, and different security policies needed throughout an AmI application. These models may be of different nature: active models, formal specifications, or ontology.

In the following we describe in more detail the two infrastructure layers of the SAMBA architecture.

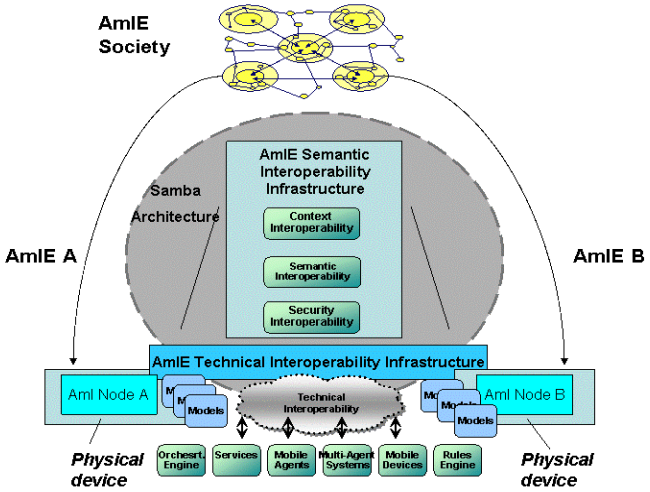

Figure 1: SAMBA Architecture

4 AmIE Technical Interoperability Infrastructures

The AmIE Technical Interoperability Infrastructure (lower layer of Figure 1) views all AmIE as *agents*, equipped with some *reasoning and dialogue capabilities*, providing individual strategies selection. This infrastructure integrates in a modular

way the service-oriented components available from the AmIE Semantic Interoperability Infrastructure (upper layer of Figure 1) and any identified autonomous agent platform.

Agents. AmI elements are represented as agents. Possible platforms that will support agents' development will be identified along with the technical issues referring to the needs for message exchanging between agents residing in different platforms. The JADE platform with the LEAP lightweight extension is an example that could satisfy our requirements, since they are open-source, widely used and compliant with the FIPA standard. The message delivery mechanism will be FIPA compliant and also support lightweight devices and mobile agents. Being FIPA compliant gives the possibility to integrate other modules or agents developed in different contexts or to easily increase the system functionality.

Agents' reasoning. An emerging area of the agent technology is the use of argumentation for the decision making of agents. Argumentation can be abstractly defined as the principled interaction of different, potentially conflicting arguments, for the sake of arriving at a consistent conclusion (see e.g. [21]). The nature of the "conclusion" can be anything, ranging from a proposition to believe, to a goal to try to achieve, to a value to try to promote. A single agent may use argumentation techniques to perform its individual reasoning because it needs to make decisions under complex preferences policies, in a highly dynamic environment (see e.g. [13]). The Gorgias open source tool offers the possibility of using argumentation for the decision making process of our agents.

Social Interactions / Autonomous Agent Dialogues. The task of modelling agent dialogues has proved to be of great importance in representing complex agent interactions. Since the work of Walton and Krabbe [23] proposing a classification of possible atomic dialogue types (i.e. deliberation, negotiation, persuasion, information-inquiry, information-seeking, eristic) a lot of works have been devoted to modelling the first five of them, the sixth being considered inappropriate in a multi-agent context. Recently, some of this work has adopted an argumentation-based approach for such dialogue modelling as can be found for example in [19, 14]. However, there exist only a few cases (see e.g. [19, 17]) of study of the combination of atomic dialogues and of the particular combination of embedded dialogues [11]. The innovation of SAMBA, in this field, is the realisation of the different dialogue types (i.e. negotiation, persuasion, deliberation, info-seeking, info-inquiry) identified in the literature using the two-level rules scheme [15] and enabling the agents that are involved in a conversation to change the type of the dialogue dynamically depending on their goals and the context the dialogue is taking place.

AmIE Technical Interoperability Infrastructure. This infrastructure will support agent's actions and decisions, and mechanisms favouring the use of service-oriented components by agents. The link with services available from the AmIE Semantic Interoperability Infrastructure is provided by a series of executable models used at run-time to invoke the needed services, thus supporting multi-technology integrations (Web services, mobile agents, MAS).

5 AmIE Semantic Interoperability Infrastructure

The goal of the AmIE Semantic Interoperability Infrastructure is to facilitate interoperability and interactions/communication among the diverse AmI elements. The resulting layer will span across the various heterogeneous, existing and emerging, service- and agent- related technologies and standards (WSDL, OWL-S, WSMO, UDDI, JXTA, FIPA, LARKS, etc.) and will provide a means for the *enactment of functional and semantic interoperability* among the AmI elements. AmI elements in SAMBA (being either agents or heterogeneous services,) expose their functionality and semantics through the use of appropriate description documents and interfaces. The provided layer will specify what these descriptions and interfaces comprise, and how they are used for the purposes of publication, discovery and invocation of the respective agents/services.

Models and Languages. This infrastructure will rely on the definition of the three following elements: 1. a *generic AmI element model* supporting all concepts needed for the description of the functional and non-functional behaviour of AmI elements within a service-oriented context; 2. *an Ambient Intelligence Element Description Language* for the AmI elements to publish their specifications or requests; 2. an *Ambient Intelligence Element Query Language* supporting the unified formulation of requests used by AmI elements for discovering each other, based on their published descriptions.

AmIE Semantic Interoperability Infrastructure. Each AmI element in SAMBA must be able to perform the following: 1. advertise its functionality; 2.discover other AmI elements that implement/offer a specific functionality; 3. interact with other AmI elements, in terms of accessing and invoking their offered functionality. The AmIE Semantic Interoperability Infrastructure, acting as a service-oriented middleware, will provide the seamless, transparent accomplishment of the above tasks at runtime. This engine will implement the AmIEQL language and will be used by AmI elements for the purposes of agent and service discovery. It will employ ontology-based techniques and mechanisms to enhance the discovery process, while the implementation of a matchmaking algorithm will ensure the compliance of query results with the initial requirements. Moreover, interactions with external registries and networks will take place in a seamless, transparent manner. Thus, a wide number of existing and emerging registry types (e.g. UDDI, LDAP, ebXML, etc.) will be accessible for discovering existing, registered SAMBA AmI elements. This infrastructure will also provide the necessary mechanisms, in order to allow the publication of service descriptions to a registry of choice, as well as the invocation of heterogeneous services (Web, P2P services), as part of the interaction among AmI elements. Invocation of agents will be catered by the underlying AmI Technical Interoperability Infrastructure. Like discovery, both publication and service invocation will take place in a unified, transparent manner. Thus, the overall platform will retain interoperability among heterogeneous AmI elements. This infrastructure will be open and extensible; it will cater for the integration of many diverse and heterogeneous service and agent technologies. Thus, the offered platform will render feasible their interoperability and seamless interaction.

6 AmIE MDE Executable Models

Besides the technological infrastructure supporting AmI systems, it is important as well to investigate development techniques supporting semantic interoperability of AmI elements. Semantic interoperability for autonomous elements implies independence of communication, independence of business flow, and independence of information format, or APIs descriptions. The Model Driven Architecture (MDA) is a promising approach for the support of semantic interoperability due to its promise of providing consistent models at different abstraction layers with well-defined mappings in between these layers. MDA is the OMG's instance of an approach to software development coming to be known as Model Driven Engineering (MDE) or Model Driven Development (MDD). MDE focuses on the Model as the primary artefact in the development process, with transformations as the primary activity mechanism, which is used to map information from one model to another. The model-driven approach in the SAMBA architecture will provide the following benefits:

- AmI models describing at different abstraction levels the AmI elements and interactions;
- Modelling language for expressing models used at run-time;
- Executable models and appropriate transformations.

Model-Driven Engineering Approach. The high-level models shall provide a framework that makes it easier for the designer to specify the AmI elements and their interactions than working with low-level specifications. We will specify and implement automated transformations from the AmI models to the AmI run-time environment. The modelling language will logically be separated into different parts which represent different views of AmI elements, such as component description, interface description, static information and behavioural/executable specifications. Furthermore the modelling language needs to support the semantic, functional, non-functional and security aspects of the SAMBA platform.

Transformation from AmI-models to AmI executable run-time specifications. A series of transformations starting from high-level models, via detailed models, and ending at AmI executable specifications will be provided. Orthogonal to these high-level-to-low-level transformations, we will also aim to define transformations for the different parts of the AmI modelling language (component, interface, static, behavioural) into the equivalent parts within the AmI run-time environment.

7 Related Work

AmI Supporting Infrastructures and Technology. Current existing techniques for realising AmI systems focus on isolated requirements, such as electronic device technology, user requirements, or specific aspects of the supporting infrastructure. At the infrastructure level, some approaches, which are essentially user/consumer-centred, are targeted towards offering relevant information and services to the

individual. Some rely on an underlying infrastructure that allows service discovery to be built through a component based engineering approach, or by using mobile agents for connecting AmI elements. Other approaches concentrate on the use of context tags, which capture and communicate information about the environment. At the *application level*, some other approaches take a socio-economic view, where business models underpin the AmI vision that devices have roles and identities. They are usually built on top of existing underlying technologies. Finally, yet other approaches focus on virtual organisations, co-operative workplaces in support of AmI, or on delivering particular portable audio-visual devices providing location information to disabled users. Although there are some research initiatives in the field of ambient intelligence, only a few of them focus on providing a computing infrastructure to the AmI elements seen as an ecosystem. In addition, it seems that there is no attempt at providing an integrated or systematic solution to problems such as semantic interoperability of AmI elements, service discovery and composition, and application development.

The SAMBA approach is not centred on providing specific services to users; it is centred on *providing an infrastructure for the AmI elements* to enable their interactions. It intends to investigate and derive an integrated computing infrastructure for AmI elements, allowing them to carry out their own tasks autonomously, while interacting socially and semantically with each other.

Semantic Interoperability. Formal specifications and models are more and more being used at run-time in conjunction with executing code in order to address semantic interoperability and adaptability issues. Self-configuring systems, specification-carrying code are attempts to replace traditional well-agreed (in advance) APIs with formal specifications understood at run-time by some middleware infrastructure [18]. This avoids the need of having shared ontologies, or agreed contracts, thus favouring a high-degree of interaction among heterogeneously designed components. Following the same ideas, but a larger level of granularity, B2B middleware for interoperable business applications, are addressing similar concerns: allowing interaction and run-time evolution of independently developed business applications [16]. Dynamic self-organising software architectures are also being defined in order to provide infrastructures where "components automatically configure their interactions in a way compatible with an overall architectural specification" [12]. Open issues related to semantic interoperability through the use of formal models or specifications encompass: the definition of specific languages for expressing the different functional and non-functional aspects; reasoning tool for processing specifications at run-time; and run-time infrastructure supporting the processing of these specifications and the execution of the corresponding code.

The SAMBA infrastructure addresses these issues and enhances the state of the art, by providing: *active models* for specifying different concerns at run-time (from functional to non-functional, to policies); *run-time reasoning tools* for processing those models, and a *technical infrastructure* supporting the execution of the underlying components.

Service-Oriented Architecture*s.* The current state of the art in service-oriented technology is characterized by a large number of heterogeneous services (e.g. Web services and P2P services), of different scope, origins and architecture models. All

these types of services employ different/incompatible architectural models, protocols, and standards for service description, discovery, publication and invocation. More specifically:

- ***Web services*** are modular, self-described applications that can be described and discovered as XML artefacts. They mainly offer coarse-grained business functionality and abide by the client-server model with the Web service being the server and with its users being the clients. They are accessible through the Web using established protocols namely SOAP, WSDL and UDDI. Despite the early standardization efforts for the core set of protocols, which facilitate the description, discovery and invocation of Web services, the supporting tools that are provided by vendors (e.g. IBM, SUN, Microsoft) have brought up many interoperability problems. WS-I [8] has tackled these issues with the Basic Profile v1.1 [9], which describes what web service developers should avoid in order to achieve interoperability among different web service implementations.
- ***Peer-to-peer (p2p) services*** follow the P2P architectural model. Currently there is neither any widely accepted definition of what constitutes a P2P service nor any accepted standard for their description and discovery. P2P services vary from fine-grained ones offering basic functionalities, e.g. discovery of the nodes and data in a P2P network, message exchange or network structure, to more coarse-grained ones offering high level business functionality, e.g. file sharing or instant messaging. Thus, P2P services offer useful functionality which may also be reused and contribute in service-oriented development. All protocols used for the discovery and invocation of P2P services are proprietary, e.g. JXTA, GNUTELLA.

Agents on the other hand may be regarded as autonomous, proactive and adaptive software systems that are able to collaboratively perform tasks that are assigned to them. A wide list of platforms exists today that may be used for the development of agents. FIPA [2] has provided a set of standards that promote the agent-based technology and the interoperability among heterogeneous agents. With respect to the aforementioned technologies there are many research efforts trying to integrate them. Most of these efforts use agents as a mean to facilitate the orchestration, discovery or mediation of services. Whereas, others use agents as service providers (e.g Web service providers).

SAMBA employs an approach where services and agents are regarded as distinct elements that may interoperate so as to achieve some objectives. The same approach has been applied in SODIUM [6], an EU IST project aiming to provide a platform that facilitates the unified discovery and composition of heterogeneous services. SODIUM has established a generic service model (GeSMO) [10], which incorporates the characteristics of Web, Grid and P2P services, a unified service query language (USQL) [22], which facilitates the discovery of services over heterogeneous registries and networks, as well as a unified service composition language (USCL) [20], which supports the composition of heterogeneous services.

Model Driven Engineering. The current state of the art in Model Driven Engineering (MDE) is much influenced by the ongoing standardisation activities around the OMG MDA (Model Driven Architecture) concept [5]. The approach

separates related abstraction models for the CIM (Computational Independent/Context), PIM (Platform Independent Model) and PSM (Platform Specific Model) levels. The metamodels that will be supported are supposed to be specified in the Meta Object Facility (MOF), and transformations between models will be supported by the Query View Transformation (QVT) standard language emerging from OMG. Furthermore the transformation from models to code will be supported by the MOF Model to Text (M2T) transformation language also emerging from OMG. The MODELWARE IST project is working on the creation of an environment for this [4], which is being further extended in the open source project Model Driven Development Integration, MDDI. This project is dedicated to integrate modelling tools, languages and methodologies to create fully customizable MDD environments. Lately Microsoft has also started to support the approach through their toolset for DSL – Domain Specific Languages in Visual Studio 2006. Support for mappings between different models is typically here also supported by a transformation and mapping programming, similar to the QVT standard language emerging from OMG.

The SAMBA architecture extends the model-driven approach to handle the modelling of AmI elements through a manifestation of the relevant concepts in a MOF-based meta-model.

8 Conclusions and Future Work

The SAMBA conceptual architecture offers a vision of the AmIE Society where AmIE are represented as agents and services interacting through a service-oriented middleware offering both technical and semantic interoperability based on the use of executable models.

The authors have already provided some parts of the architecture: the SODIUM infrastructure [6] provides a unified approach for composition, discovery and execution of heterogeneous services (i.e. Web, P2P, and Grid services), and the Specification-Carrying code infrastructure [18] allows interaction among components through the exclusive use of specifications. In addition, the INTEROP NoE [3] has a number of task groups investigating the model-driven approach to interoperability, with various studies of technologies for model-driven interoperability and morphosis/transformation. The INTEROP domain group for Architecture&Platforms is investigating the possibilities for a platform independent interaction model across various heterogeneous platforms, including agents, web services, grids and P2P. A group on NFA, Trust, Policies and Security is looking into interoperability of non functional aspects. The ATHENA IP [1] is realizing various model-driven tools for both semantic interoperability and integration of agents in a SOA platform. The MODELWARE IP project [4] is developing model driven tools to support an underlying open source toolset.

Upcoming projects, such as SWING [7] and MODELPLEX [4] will extend the work around semantic services and model-driven interoperability, and we hope to be in a position to later bring the various solution elements together, for a more complete realization of the SAMBA vision and architecture.

3 References

[1] ATHENA project: www.athena-ip,net
[2] FIPA – Foundation for Intelligent Physical Agents, http://www.fipa.org/
[3] INTEROP project: www.interop-noe.net
[4] MODELWARE and MODELPLEX projects: http://www.modelbased.net
[5] OMG MDA, www.omg.org/mda
[6] SODIUM: http://www.atc.gr/sodium
[7] SWING: http://www.sintef.no/content/page1883.aspx
[8] Web Services Interoperability Organization: http://www.ws-i.org
[9] WS-I Basic Profile v1.1, http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html
[10] Athanasopoulos G, Tsalgatidou A, Pantazoglou M (2006) Interoperability among Heterogeneous Services. In: Proc. of IEEE International Conference on Services Computing (SCC '06): 174-181.
[11] Dimopoulos Y, Kakas A, Moraitis, P (2005) Argumentation Based Modeling of Embedded Agent Dialogues. In: Proc. of 2nd International Workshop on Argumentation in Multi-Agent Systems, (ArgMAS'05).
[12] Georgiadis I, Magee J, Kramer J (2003) Self-organising software architectures for distributed systems. In: Wolf A, Garlan D, Kramer, J (eds) Proc. of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02): 33-38.
[13] Kakas A, Moraitis P (2003) Argumentation Based Decision Making for Autonomous Agents. In: Proc. of 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03): 883-890.
[14] Kakas A, Maudet N, Moraitis, P (2004) Layered strategies and protocols for argumentation-based agent interaction. In: Proc. of 1st International Workshop on Argumentation in Multi-Agent Systems (ArgMAS'04).
[15] Kakas A, Maudet N, Moraitis, P (2005) Modular Representation of Agent Interaction Rules through Argumentation. In: Journal of Autonomous Agents and Multi-Agent Systems, (JAAMAS), Springer US, 11(2): 189-206.
[16] Kutvonen L, Ruokolainen T, Metso J, Haataja J (2005) Interoperability middleware for federated enterprise applications in Web-Pilarcos. In: Konstantas D, Bourrières JP, Léonard M, Boudjlida N (eds) Proc. of 1st International Conference on Interoperability for Enterprise Software and Applications (I-ESA'05): 185-196.
[17] McBurney P, Parsons S (2002) Games that agents play: a formal framework for dialogues between autonomous agents. In: Journal of Logic, Language and Information, 11(3): 315-334.
[18] Oriol M, Di Marzo Serugendo G (2004) A disconnected service architecture for unanticipated run-time evolution of code. In: IEE Proceedings-Software, 151(2): 95-108.
[19] Parsons S, McBurney P, Wooldridge M (2003) The mechanics of some formal inter-agent dialogue. In: Proc. of Workshop on Agent Communication Languages: 329-348.
[20] Pautasso C, Heinis T, Alonso G (2005) D6-SODIUM Unified Service Composition Language (USCL), SODIUM project IST-FP6-004559 deliverable, June 2005.
[21] Rahwan I, Moraitis P, Reed C (eds.) (2005) Argumentation in Multi-Agent Systems: Proceedings of the First International Workshop (ArgMAS'04) LNAI, Volume 3366, Springer-Verlag, Berlin, Germany.
[22] Tsalgatidou A, Pantazoglou M, Athanasopoulos, G (2005) D8-Specification of the Unified Service Query Language (USQL), SODIUM project IST-FP6-004559 deliverable, June 2005.
[23] Walton DN, Krabbe ECW (1995) Commitment in dialogue: basic concepts of interpersonal reasoning, State University of New York Press.