

## **Building next generation Service-Oriented Architectures using argumentation agents**

Vasa Curcin<sup>1</sup>, Moustafa Ghanem<sup>2</sup>, Yike Guo<sup>2</sup>, Kostas Stathis<sup>3</sup>, Francesca Toni<sup>1</sup>

<sup>1</sup>Department of Computing, Imperial College, 180 Queen's Gate, London, SW7 2AZ, United Kingdom

<sup>2</sup>InforSense, 459a Fulham Road, London, SW10 9UZ, United Kingdom

<sup>3</sup>Department of Computing, City University, Northampton Square, London, EC1V 0HB, United Kingdom  
[vc100@doc.ic.ac.uk](mailto:vc100@doc.ic.ac.uk), [mmg@inforSense.com](mailto:mmg@inforSense.com), [yg@inforSense.com](mailto:yg@inforSense.com), [kostas@soi.city.ac.uk](mailto:kostas@soi.city.ac.uk), [ft@doc.ic.ac.uk](mailto:ft@doc.ic.ac.uk)

keywords: service-oriented architectures, agents, grid, workflows, argumentation

**Motivated by the increased bioinformatics service-based offering on the Grid, we envisage a new model for programming the Grid at a semantic, knowledge-based level of abstraction through the use of argumentative agent technology. In this setting, agents are associated with roles of service requestors, service providers and brokers. This new model provides an enhanced service composition model for grid applications, whereby service requestors, providers and brokers can profit from the dynamic composition of Grid resources and services into executable workflows. This model would thus have considerable impact on existing business practices, by enhancing the role of Grid in current business applications. Within this model, argumentative agents are equipped with methods for negotiation with other agents. Agents act on behalf of their owners, be it in the consumer, provider or broker roles, or a combination thereof. With the support of argumentation processes, they decide how to fulfill the demands of their owners by creating, managing and joining virtual organisations, understood as societies of agents, while pursuing their own goals.**

## 1. Introduction

In the past decade, under the influence of the World Wide Web, many software tools have evolved from standalone separate computer programs into locally managed, globally accessible applications, offered within highly distributed environments, such as the Grid. The traditional Service Oriented Architecture (SOA) view offers a model for orchestrating these tools into complex applications by means of workflow technology. By participating in these applications, service providers form Virtual Organizations (VOs) that are assembled and disassembled sometimes even for the needs of one single agent, if the price is right.

However, no clear means are specified for encapsulating and fulfilling requirements from service requesters and, especially, service providers. For example, the provider cannot implement policies on which service they prefer to cooperate with and implement different priorities for different classes of users. Also, workflows produced in this manner are statically bound to service instances and not responsive to changes in the environment, such as a certain service changing its *modus operandi* or simply going down. Finally, the metadata associated with the services in the established protocols such as WSDL, SOAP and UDDI, offers little beyond purely technical connectivity information, and nothing in manner of how services can be composed in meaningful manners. This last issue is now being investigated within the Semantic Web initiative, by using domain and service ontologies as the basis for composition. Because of this, it is difficult for complex business models, involving partners, preferences, and commitments, to be implemented within the SOA.

### 1.1 Agent technology for Service Composition

The use of agent technology offers a powerful solution to dynamic service composition in a distributed setting. Different services can be associated with autonomous agents that can negotiate, on behalf of service requesters and providers, implementation plans that take into account the preferences of both sides.

While this idea has been investigated before, it has not previously been explored fully to provide a working system. For example, Foster et al [1] describe this relationship between service oriented grid computing and agent technology as one between “brawn” and “brain”. Service-oriented grid computing is concerned with sharing, selection, and aggregation of distributed resources and services and is focused on the development of middleware and tools to hide the heterogeneity of the underlying resources and services and to guarantee their integration. On the other hand, agent computing research is concerned with the development of concepts, methodologies, and algorithms for autonomous problem solvers that can act and cooperate flexibly in uncertain and dynamic environments in order to achieve their aims and objectives.

In this paper we propose the use of agents that use argumentation-based techniques in order to decide their actions and cooperate by means of negotiation. Our agent-based approach differs from other agent-based approaches in the literature, e.g. [17], in that, although we advocate service mediation for dynamic composition through agents, we do this from the perspective of Grid-based Virtual Organizations. While we still rely on P2P, it is purely a communication mechanism rather than the core design. In a similar manner, we will use existing ontologies and reasoning mechanisms to construct agents’ knowledge bases.

### 1.2 Key Challenges

Foster et al [1] describe a broad set of key challenges that need to be addressed for successfully using agent technology for service composition. These can be summarized as providing a base set of architectural elements and methods for:

1. Creation, re-use and composition of interoperable components/services.
2. Large-scale system management.

3. Describing, discovering and composing services.
4. Enabling semantic integration of components and services
5. Expressing inter-agent negotiation
6. Formation and management of Virtual Organizations
7. Expressing, reasoning about and negotiating using trust
8. Expressing and reasoning about system predictability (performance and safety)
9. Enabling human-computer collaboration
10. Definition of metrics for system evaluation

In this paper, we use a real-world bioinformatics example to illustrate how agents can be used to address some of these challenges and highlight other challenges that need to be met. In particular we use our work on the Discovery Net system [2] as a basis for addressing challenges 1-3 above and propose an extended framework that addresses challenges 4-8.

The architecture presented in this paper is concerned with defining the anatomy of and interactions amongst argumentation agents that enables the full mapping of the real-world organization strategies and aims to agents representing them in VOs. As such, our planning architecture is open to different application domains as well as multiple implementations, which may include such features as enterprise-level security, privacy, authentication, accounting of service costs, and semantic annotation. The architecture does not presume or require any standards for these features, and they are left entirely for the implementers to decide on. For example, one implementation option can be based on using the GRIA middleware [18] which provides support for authentication mechanisms and an accounting/QoS model of services using industrial standards for web service computing, while easily interfacing with the negotiation procedures and decision-making behavior of agents..

### **1.3 Paper Layout**

The remainder of this paper is organized as follows: Background to Discovery Net is presented in section 2. The motivating use case from the field of bioinformatics is introduced in section 3. Section 4 describes the standard workflow solution, and we present both the solutions and the problems it offers. Agents solve some of these problems, most prominently dynamic composition, as described in section 5, but not all. The introduction of VOs allows us to formalize the cooperation between the participating agents, as described in section 6, but new problems arise, e.g. related to creation of VOs. In section 7 we introduce argumentation agents as a possible mechanism for building a solution to these issues. Finally, section 8 summarizes the findings and main contributions of the paper.

## **2. The Discovery Net Infrastructure**

The Discovery Net infrastructure has been designed and implemented based on a workflow model, allowing the composition of data analysis services and resources declared as web/grid services. Informally, a workflow (see Figure 1) is an abstract description of the steps required for executing a particular real-world process, and the flow of information between these tasks. Work passes through the flow from start to finish and activities might be executed by humans or by system functions.

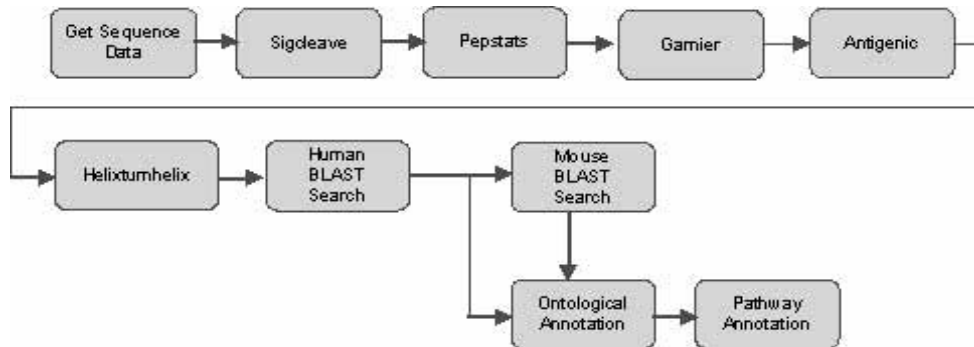


Figure 1: Abstract workflow for sequence annotation data analysis.

Within Discovery Net, computational components are treated as remote services or black boxes with known input and output interfaces described using web service protocols. Each component has a Service Descriptor that provides a description of the input and output ports of the component, the type of data that can be passed to the component, and parameters of the service that a user might want to change. Once the service descriptor has been developed, the component can be added into the Discovery Net by registering with the component service. This will dynamically make the service available to clients so that users can take advantage of the new service.

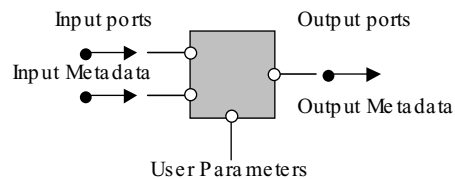


Figure 2: The anatomy of a Discovery Net component.

The abstraction is shown in Figure 2, depicting the three main types of interface. The Output port interface describes what will be output by the service at runtime. The Input port interfaces describe the required inputs to the service. These can be constrained to only accept connections from output ports that provide specific metadata. This is important for describing valid discovery processes according to the composition language and to help users who compose services together into a specific application build meaningful processes. The final interface is the parameterization, that the user is allowed to instantiate to customise the service.

Once designed, a workflow can be submitted for execution by an execution engine that controls the invocation and data transfer between the different components. The secure execution of the individual components themselves over high performance computing resources is achieved using Grid computing protocols [5]. The coordination of the execution of such services is described as workflows expressed in DPML [14], an XML-based workflow language. The execution management of the distributed workflows is handled by Discovery Net's workflow execution engine [13].

### 3. Bioinformatics use case

In order to describe the issues relevant to the service provision on the Grid, we introduce a small example that mirrors some common tasks in the bioinformatics research today. While the steps have been simplified so as to focus on the main issues, they are still valid and accurately reflect real-life.

The user, working at a disease research institute, has a protein sequence obtained from a human patient with a novel form of a certain disease. In order to gain some understanding of the behaviour of that protein, the sequence needs to be annotated with its physical properties and compared to other similar sequences to see what common functionality it may have with them. First of all, the sequence is run through a series of structure prediction services such as sigcleave (prediction of signal cleavage sites), pepstats (peptide statistics), garnier (secondary structure prediction), antigenic (prediction of potentially antigenic regions) and helixturnhelix (tertiary structure prediction). In the next step, a BLAST similarity search is conducted against the database of human proteins. Sequences are split into those that have good matches, poor matches and no matches. Poor and no matches are then passed to be queried against a similar database of mouse sequences. Good matches are extracted, combined with the previous ones, aligned and annotated with ontological and pathway information, which will provide the information on the sequence functions and the processes it partakes in within the cell. The resulting information is all sent back to the user for further manual analysis.

An obvious problem with the aforementioned example is that the user does not necessarily have access to all these tools and data sources. The structure prediction tools listed are part of the EMBOSS sequence analysis application suite, and are often found exported as web services on research institute web pages and intranet servers within companies. Human and mouse protein databases are located at the European Bioinformatics Institute (EBI), the National Center for Bioinformatics (NCBI) and other locations, and also on local servers in some organizations. Ontological annotation is most commonly performed from the Gene Ontology (GO) website, or from a local copy. Finally, pathway information requested can be obtained from the Kyoto Encyclopedia of Genes and Genomes (KEGG) website.

Retrieving pure data is not necessarily a problem for such a small task. Equipped with a web browser, copy-paste functionality, and a significant amount of patience, the user can retrieve a bunch of HTML pages. However, extracting useful information from that raw data and structuring it in such a way that it is useful to the scientist is the first major challenge, and the one that workflow technologies have been introduced to solve.

### 4. Workflow view

Introducing a workflow based infrastructure provides a structured view of the task, with data dependencies and interactions between individual components clearly specified. The workflow can even be defined in an abstract manner, only specifying service descriptions, leaving the selection of actual services to a broker with access to service registries. In that way, a separation is achieved between the requestor and provider, and the user does not have to search manually for the providers who can fulfil his request.

The conventional mapping of abstract workflows into concrete workflows (without agent technologies) can be seen in more detail in the figure below:

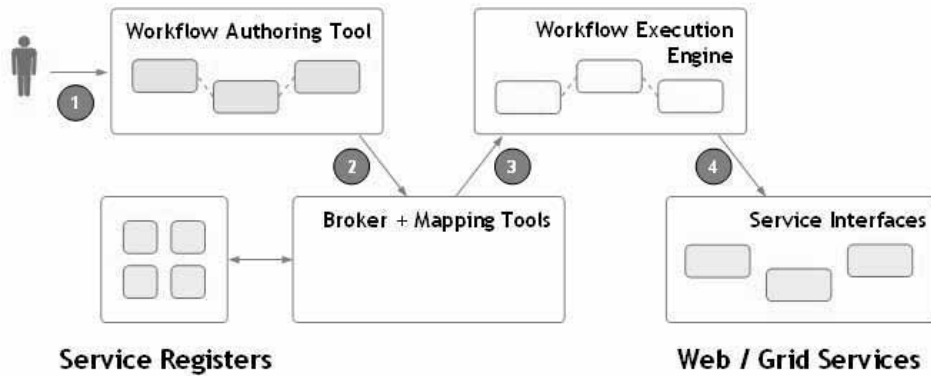


Figure 3: Mapping abstract workflows into concrete workflows without agent technologies

Here, Service Registers store useful information about service locations, capabilities, and so on. The system goes through the following phases:

1. **Requirements specification.** Users describe their task as an abstract workflow together with a set of requirements on services.
2. **Broker matching.** Broker uses a match-making algorithm to find services matching user requirement. Mapping tools transform abstract representation of workflows into executable ones.
3. **Execution.** Workflow engine manages the execution of the executable workflow by invoking remote services, handling data transfers, etc.

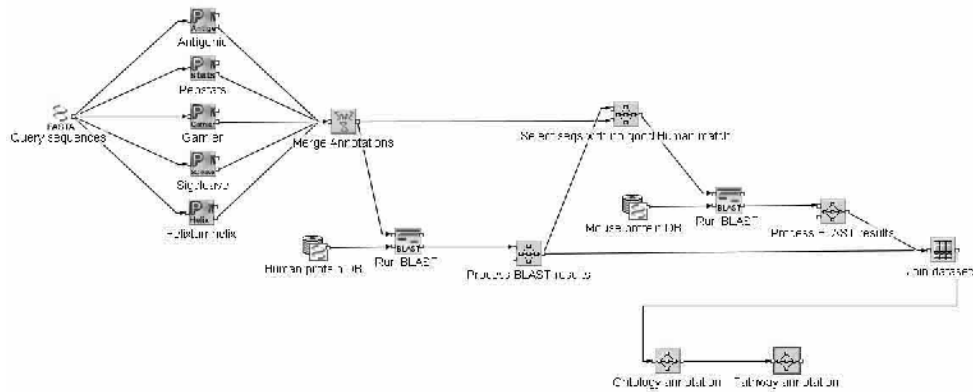


Figure 4: Workflow in Discovery Net

This solution leaves a number of issues open. In particular, service providers are still ignorant of whom they are providing the service to, and their composition is handled by the broker. Also, the execution plan, once created by the broker and returned to the user, is static and cannot change even if a service malfunctions. Commitments between actors in the process are, by necessity, trivial since there is no way of

dynamically creating them and representing them. Finally, service providers are passive participants in the workflow, with no autonomy and no method of creating a business model for operation.

## 5. Agent view

Agents are computational entities with the properties of being:

- proactive - towards achieving goals and objectives they hold ;
- situated in the environment in which they are located - thus being aware of any changes that might occur in such environment ;
- reactive - to changes in the environment ;
- social - aware of the existence of other agents in their environment and capable of interacting with them, e.g. by communication and negotiation.

In general, agents can be seen as the natural progression of distributed artificial intelligence on one hand, and object-oriented software engineering on the other.

Agents have recently been recognized as a useful extension to the Grid [1], acting as representatives for Grid participants who provide or require access to resources/services. The agents' usefulness on the grid is due to a number of features they commonly exhibit. Indeed, agents provide autonomous problem solving capabilities in dynamically changing environments, are capable of autonomous interaction and negotiation with one another, and can process "semantic" information, typically held within their internal state, and representing their beliefs, goals, plans, utilities and preferences. Meta-data on contents and features of resources/services can be represented within the internal state of these agents. Agents can thus reason on behalf of service providers and requestors and seek situations, which maximize their profit. Thereby, they introduce a mechanism by which an organization providing the service can implement its private policies, so as to reflect their business model. Note that these policies are not necessarily advertised in service registries.

Expanding our earlier use case to include both workflow infrastructure and agent-based services, the user would now describe the steps required using a workflow description language and submit that abstract description to a broker which would, using service/agent registries, find suitable agents providing the required services and construct an initial workflow. The workflow in our example consists of linking first to Imperial College's BioGrid agent which offers structural annotations. Both BLAST sequence searches are provided by NCBI and the broker finds their price and execution estimate within the specified criteria. A Clustalw sequence alignment service is offered by an autonomous agent who is registered with the broker's registry. Finally, the functional annotations are offered by a MyGrid agent running at the University of Manchester.

Once all agents have been selected and agreed, the execution scenario is returned to the broker (encompassing the generic data filtering tasks), who then has to find the appropriate execution engine. The schema details given for the workflow are used to assemble a requirements description. Since the workflow is relatively simple, many engines can execute it. However, the user's organization has an account with InforSense Execution Provision, and the service is free for them, so the broker selects that execution engine.

Thus, agents can help with many of the challenges of the workflow solution, e.g. by searching for providers among themselves, perform task decomposition, and represent the interests of service providers, requestors and brokers. In order to allow agents to support service composition though a number of challenges need to be met, notably:

- ◊ the representation of messages,
- ◊ the specification and realisation of communication protocols and policies,
- ◊ the specification and realisation of the reasoning internal to the agent, taking into account preferences and goals,

- ◇ the realisation of means for problem decomposition, within and across agents, and
- ◇ the representation of “contracts” committing the agents to the agreed workflow .

## 6. Virtual Organization View

Many of the challenges of the agent solution can be addressed by realizing VOs, regulating to some extent the interactions amongst some of the agents and providing standard protocols and policies. VOs can be understood as societies of agents, an abstraction more complex than the pure agent one. An agent society is a collection of agents whose interactions are regulated by social orders and contracts specifying the agent roles, commitments, and positions in the society, as well as the modalities of their interactions. VOs in the Grid [4] can be viewed as special instances of agent societies connected through a communication network in order to share skills to achieve some overall (set of) objectives and regulate the interaction amongst agents. The problem of sharing resources/services in VOs then becomes the problem of negotiating resources/services within agent societies.

In the case of our use case and the concrete workflow given in the earlier section, the VO consists of the Imperial College’s BioGrid agent, the NCBI, the agent providing the Clustalw sequence alignment service, and the MyGrid agent, all committed to the concrete workflow of execution.

The following figure illustrates the relationship between societies of agents (corresponding to VOs) in ArguGRID and service-oriented computing:

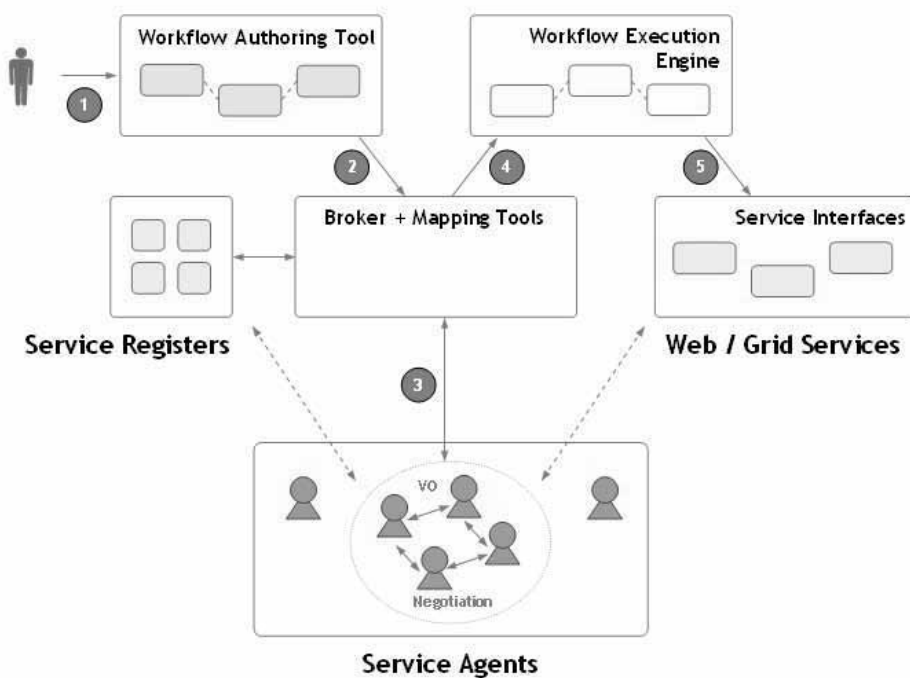


Figure 5: Mapping abstract workflows into concrete workflows

In Figure 5, step 3 represents the interfacing of Broker with agents to retrieve the workflow implementation plan that they dynamically compute. The Broker then uses mapping tools to transform the implementation plan into an executable workflow. Step 3 effectively becomes the interface between the service-oriented world and the agents. Once the concrete workflow is successfully executed, at step 5, the VO is dissolved. Until then, the agents providing/requiring services according to the workflow commit to it and are bound by a “contract”.

The following figure illustrates the relationship between agents and services.

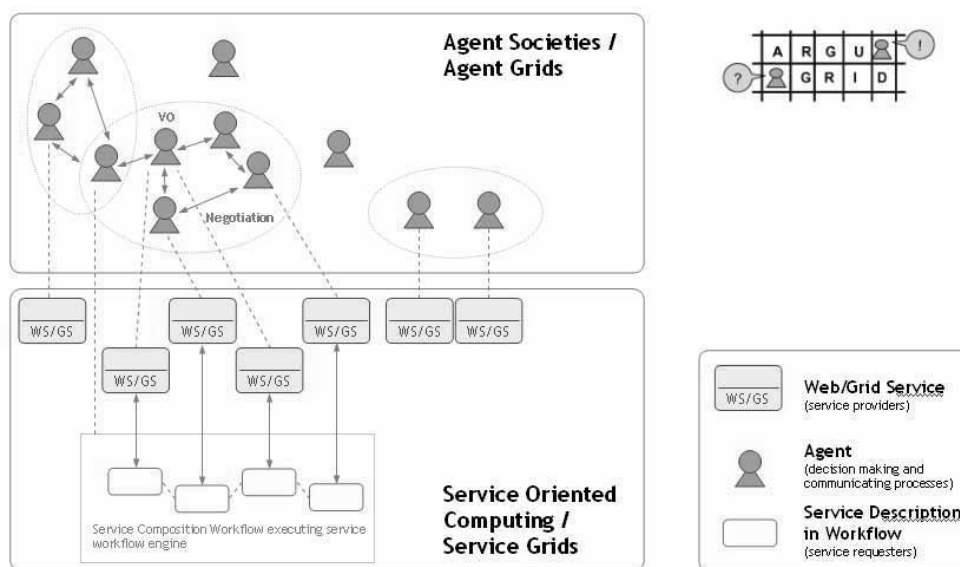


Figure 6: Agent-based service environment

Going back to our example again, let us say the BioGrid agent does not perform the calculations itself, but it outsources them to a wide number of organizations that offer EMBOSS and GCG tools as web services. The business value of the BioGrid agent in this scenario is that it actively collates and maintains the information about available sequence information providers together with their costs and other metadata. It also performs the negotiation with the other agents to build an offering that satisfies the requirements presented to it and that will satisfy its own goals. Consequently, it takes the responsibility for their performance, and put its trust rating at risk.

We can also assume that the MyGrid provision model works in a similar way to BioGrid, by maintaining a registry of annotation sources and being able to obtain the best ‘bargains’. In our case, it will select the KEGG database to retrieve the pathway information and GeneOntology service for ontology data.

In this setting, trust [3] and reputation also play a fundamental role in reducing the uncertainty in decision making of agents in a distributed system whose participants have a high degree of freedom [3]. Intuitively, trusting an agent means believing that it will do what it says given an opportunity to defect to get a better outcome, which is an important element in deciding the configuration of VOs. Also, the notion of trust is not necessarily transitive (even though requests may be), so if agent A trusts agent B, and agent B trusts agent C, this does not have to mean that agent A trusts agent C. In addition, the contract terms between B and C may be significantly different than what would C offer to A (e.g. due to some previous contracts).

It is important to equip VOs with mechanisms to deal with situations such as violation of trust, when an agent that committed to provide a service defaults on that commitment, e.g. a helixturnhelix provider that BioGrid found violating its contract. The challenges that need to be met for providing such mechanisms in VOs include the following:

- ◊ Would the agent require some sort of approval from its requestor before swapping the violating service for another one?
- ◊ Would there be cost compensation involved?
- ◊ And how are the initial contracts negotiated?

## 7. Argumentation agents as the path to solution

The key point missing from the previous architectures is that agents still require some formal means for decision-making and for negotiating with one another during the creation, maintenance and dissolution of VOs. In our solution, we propose enriching the agent-based services with argumentative capabilities to support their decision-making within the negotiation process. Trust plays a role in the decision-making and negotiation. It is this enhanced power during the negotiation process that enables a fuller modeling of the agent owner's business needs. Agendas, both public and private, strategies, preferred partnerships and the like can be formally represented and encoded in the agent itself, minimizing the need for human involvement – a necessary step if the agent technology is to scale beyond trivial virtual examples. More importantly, argumentative agents can reason and take decisions, on behalf of users and providers, weighting various arguments pro and against decisions against one another and taking into account the needs, strategies etc of users and providers. These argumentative reasoning capabilities distinguish our agents from those developed by others, e.g. [17], focusing instead on lower-level architectural features of agents. The agents' decisions amount to:

- ◊ providing services and at which conditions (time, price, etc)
- ◊ engaging with other agents within VOs
- ◊ trusting other agents
- ◊ revising earlier agreements and at which conditions
- ◊ replying to the requests of other agents.

Argumentation, simply stated, focuses on interactions where parties plead for and against some conclusion or decision [5]. In its most abstract form, an argumentation system is simply a set of arguments and a binary relation representing the attack-relation between the arguments [6]. By instantiating the notion of arguments and the attack relations, different argument systems can be constructed [6]. In contrast to traditional approaches to decision making that are based on numerical values, argument-based decision making is based on logic. Several models, computational frameworks and implementations for both qualitative and quantitative argumentative inferences have been developed in the literature [8]. Moreover, several tools exist for argumentation-based collaborative decision making [9]. Argumentation also provides a natural framework for studying the semantic foundations of dialogues, which can be viewed as a declarative abstraction of the interaction protocols of agents in a VO. There are several types of relevant dialogues: information seeking, inquiry, persuasion and negotiation dialogue. Several dialogues systems are available [12].

Because of the above features, argumentation models and scenarios have recently emerged as influential within the agent community, as witnessed by a significant workshop series created to investigate the area at the intersection of Argumentation and Multi-Agent Systems (ArgMas). Indeed, argumentation provides a powerful framework for interacting agents that need to reason to make decisions, assess the validity of information they become aware of, or resolving conflicts and differences of opinion. It is an essential ingredient of inter-agent dialogue, negotiation persuasion and collaborative decision-making [15]. Argumentation can thus serve as a unifying medium to provide a model for agent-based semantic grid systems, in that it can support:

- reasoning and decision-making process of agents controlling/requiring access to resources/services in the grid,
- inter-agent negotiation process, to agree on shared and coordinated access to resources and services,
- identification of needs and formation phases in the life of a VO,
- definition of workflows and contracts characterizing VOs, derived at run-time from partial specifications provided from the different parties involved in the VOs,
- resolution of disputes and disagreements amongst agents and VOs, with respect to contracts in VOs,
- identification of (support for) useful information about the trust level of agents and VOs.

The architecture for argumentative agents we propose is depicted in Figure 8 below.

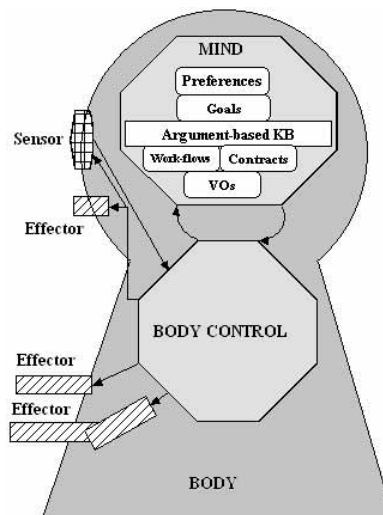


Figure 8: The mind-body architecture of argumentative agents

This architecture is adapted from the mind-body architecture of PROSOCS agents [16]. The mind uses an argumentation based Knowledge Base (KB), containing amongst others, the representation of workflows, contracts, and knowledge about the VOs. The mind also makes decisions about actions according to preferences and goals, which mirror the business strategies and aims of the entity that the agent is representing in the VO.

The agent accesses the environment in which it is situated by obtaining observations from sensors. These change the knowledge of the agent and may include observations from communications with other agents. Decisions of the agent are executed by the body, via effectors, as actions in the environment in which the agent is situated. Speech acts uttered by the agent itself are generated in this way.

One key consequence of this solution is that all participants in the SOA become agents. Requestor, broker and provider are merely roles that they play in any particular VO. Argumentation becomes the tool for the internal decision-making process of agents and the basis for the peer-to-peer negotiation between the agents to agree service implementations and service composition plans. The following simple abstract example illustrates both uses of argumentation in the service setting.

### 7.1 An example of argumentative agents for service composition

Assume the user requires an abstract workflow whereby some service A followed by some other service B are needed. In the abstract workflow, services are specified in terms of generic functionalities, possibly via some code agreed within the system. For example, in our use case, A may be Human BLAST and B may be Mouse BLAST. The user is a bioinformatician working at European Bioinformatics Institute (EBI) and the broker is EBI's default broker.

All requirements on the abstract workflow will be passed by the user to a user agent supporting the user itself during the negotiation procedure with other agents, aiming at rendering the abstract workflow "concrete" by identifying concrete service providers, and fulfilling all constraints on the services imposed by the user, as known by the user agent. These constraints may represent, in general:

- some temporal deadlines, e.g. that B needs to be provided by a certain date/time, and A needs to be started after a certain date/time
- some upper bound on cost, e.g. that the user is prepared to pay only up to a maximum amount
- some lower bound on some quality of service parameters, e.g. that data needs to be transferred at some given speed per second
- some preference on service providers, e.g. that the user would not consider some specific provider (maybe due to some past bad experience with that provider).

The abstract request given earlier is submitted by the user agent to a broker agent, e.g. via a message that we can abstractly represent as follows:

*msg(user\_agent, broker\_agent, user\_agent:Constraints:workflow(A,B), T1)*

where *T1* is the time the request is posted by the user agent to the broker agent and *Constraints* will in general be a subset of the ones specified by the user. Broker agents are not strictly speaking necessary, but are a useful addition. *Constraints* may be empty, in case the user does not want any of its preferences to be disclosed to the outside world. Assume for this simple example that the communicated *Constraints* are only temporal, but the user agent also has an upper bound for the cost of the implementation of the workflow that it does not disclose.

Then, the agent acting as the broker may pass the original request to some concrete service provider it believes might be able to help the user, e.g. by means of a message:

*msg(broker\_agent, service\_agent, user\_agent:Rating:Constraints:workflow(A,B), T2)*

where *Rating* might be a numerical value evaluating the reliability of the user agent as known to the broker agent, or the level of trust the broker agent attributes to the user agent. Then, the service agent may be reasoning internally by means of argumentation, to decide amongst two possible instantiations of the request, both satisfying the given *Constraints*:

- $S_A$  and  $S_B$ , for some *Cost*
- $R_A$  and  $R_B$ , for some *LowerCost*

. Indeed, the service agent may hold internal beliefs that  $S_A$  and  $S_B$  together result in the functionalities required for A and B (Similarly for  $R_A$  and  $R_B$ ). These internal beliefs may also allow the computation of the *Cost*. Within argumentative agents, these beliefs "link together" to provide an argument. Given that there are arguments for  $S_A$ ,  $S_B$  and  $R_A$ ,  $R_B$ , and that these arguments are in conflict, the argumentative reasoning of the agent might select the first argument. This may be achieved by using a general preference rule of the service agent to always try and maximize its profit (thus choosing the argument giving the higher *Cost*). This reasoning allows to single out one possible argument as preferred over the conflicting one. If a different preference rule existed, e.g. that customers with a high *Rating* should be given the best

*LowerCost*, then, given that the user agent's *Rating* is high, the alternative argument would have been selected.

After the (first) decision has been taken, this can be notified by means of a message

```
msg(service_agent, broker_agent,  
     service_agent:user_agent:Cost:workflow(services(SA, SB)), T3)
```

which is passed to the user agent as:

```
msg(broker_agent, user_agent,  
     service_agent:user_agent:NewRating:workflow(services(SA, SB)), T4)
```

where *NewRating* is a numerical value evaluating the reliability of the service agent as known to the broker agent. Assume this reliability is high. The user agent may however evaluate this offer as too expensive, and notify a rejection supported by an argument to the reasons of this rejection, e.g. by a message

```
msg(user_agent service_agent,  
     reject(service_agent:user_agent:Cost:workflow(services(SA, SB))  
           argument(high(Cost)), T5)
```

Then, the service agent might decide to lower the *Cost*, taking into account the high rating of the user agent, or might decide to offer the alternative instantiation of the workflow with the *LowerCost*, depending on its internal preferences guiding the argumentative decision-making process. It might then decide to go for a counter-offer that the user agent may now accept, even if the proposed new cost might still be higher than originally foreseen.

In this case, although a given requirement cannot be satisfied (an upper bound on the cost), negotiation, taking into account the agents' preferences and willingness to compromise, may lead to a solution. In this instance the user/user agent may be prepared to go for a higher cost than originally envisaged due to the fact that rating of the service provider is high.

## 7.2 Challenges

In general, this process might involve any number of agents of any kind. Users might want to change their abstract workflows dynamically, and their user agents might thus be engaged in dynamic negotiation whereby, e.g., the order of services might be changed to take into account special offers or good combinations of services as emerged during negotiation. Also, service providers may join together and provide combined services at advantageous costs, and user agents may use this information for decision making during negotiation. These combinations might result in VOs amongst agents, which may be dynamically enlarged during negotiation with user agents.

Once an abstract workflow is rendered concrete, the participants in the resulting concrete virtual organisation are bound by a contract, specifying the temporal constraints on the provision of services, any costs and compensations agreed during negotiation, and conditions for the dissolution of the contract. Contracts need to be represented in a way that all agents can refer to them. During the life-time of VOs, disputes might emerge concerning the fulfilment of obligations to the contracts, and argumentation will in general be useful also for the resolution of these disputes.

## 8. Summary and conclusions

In this paper, we described how the problems facing the next generation of Service Oriented Architectures can be solved by agent-based services using argumentation as the means of decision-making and negotiation between them.

A set of challenges was isolated to reflect the complexities of a mature setting with each participant having their own agendas and preferences. The architecture introduced to address these is scalable enough to handle the heterogeneity of various services, policies from multiple providers and complex contracts specifying different agents' obligations.

In our view argumentation-based agent communities are the next step in the development of service architectures. Agents are evolving beyond just being ignorant providers/requestors into entities that are acting as true representatives of their parent organization/individual on the Grid. As such, they will be the key element in establishing Grid service economies by internalizing composition reasoning and collaborator selection within their logic rather than imposing them from the outside.

## 9. Acknowledgments

This work was partially funded by the Sixth Framework IST programme of the EC, under the 035200 ARGUGRID project. The last author has also been supported by a UK Royal Academy of Engineering/Leverhulme Trust senior research fellowship.

## 10. References

- [1] I. Foster, N. Jennings, C. Kesselman. "Brain meets brawn: why grid and agents need each other". Proc. AAMAS 2004.
- [2] V. Curcin, M. Ghanem, Y. Guo, M. Kohler, A Rowe, P. Wendel. "Discovery Net: Towards a Grid of Knowledge Discovery". *ACM KDD-2002*. July 2002 Edmonton, Canada.
- [3] J. Patel, W.T.L. Teacy, N.R. Jennings, M. Luck, S. Chalmers, N. Oren, T.J. Norman, A. Preece, P.M.D. Gray, G. Shercliff, P.J. Stockreisser, J. Shao, W. Gray, N.J. Fiddian, S. Thompson. "Monitoring, Policing and Trust for Grid-Based Virtual Organisation"s. In Proceedings of the UK e-Science All Hands Meeting 2005, Nottingham UK.
- [4] J. Patel, W.T. L. Teacy, N.R. Jennings, M. Luck, S. Chalmers, N. Oren, T.J. Norman, A. Preece, P. M. D. Gray, G. Shercliff, P.J. Stockreisser, J. Shao, W.A. Gray, N.J. Fiddian, S. Thompson. "Agent-Based Virtual Organisations for the Grid. *Int. Journal of Multi-Agent and Grid Systems*" 1(4) pp. 237-249. 2005
- [5] C. Chesnevar, A. Maguitman, R. Loui. "Logical Models of Argumen". *ACM Computing Surveys*, 2000.
- [6] A. Bondarenko, P.M. Dung, R.A. Kowalski, F. Toni. "An abstract, argumentation-theoretic approach to default reasoning". *Artificial Intelligence* 93, 63-101, 1997
- [7] N. Giannadakis, A. Rowe, M. Ghanem M, and Y. Guo Y. "InfoGrid: Providing Information Integration for Knowledge Discovery". *Information Science*, 2003; 3:199-226.
- [8] , P.M. Dung, R.A. Kowalski, F. Toni. "Dialectic proof procedures for assumption-based admissible argumentation". *Journal of Artificial Intelligence* (2006).
- [9] N. Karacapilidis et al. "Computer supported argumentation and collaborative decision making: The hermes system". *Information Systems*, 26, 2001
- [10] R. Haenni et al. "An Interactive Tool for probabilistic argumentative reasoning". Proc.ECSQARU 2003.
- [11] D W Glasspool et al. " Interactive decision support for medical planning". Proc AI in Medicine (AIME 2003)
- [12] A. Artikis et al. "An executable specification of an argumentation protocol". Proc of the 9th Int. Conf on AI and Law, 2003
- [13] A. Rowe, D. Kalaitzopoulos, M. Osmond, M Ghanem, Y. Guo. "The Discovery Net System for High Throughput". *Bioinformatics*. *Bioinformatics* 2003: 225-231
- [14] J. Syed J, Y. Guo Y, M. Ghanem. "Discovery Processes: Representation And Re-Use", UK *e-Science All Hands Meeting*, Sheffield UK, September, 2002.
- [15] S Parsons, C Sierra, N R Jennings: "Agents that reason and negotiate by arguing", *Journal of Logic and Computation*, 1998
- [16] K. Stathis, A. Kakas, W. Lu, N. Demetriou, U. Endriss, A. Bracciali, PROSOCS: A Platform for building Software Agents in Computational Logic. In proceedings of the 4th International Symposium from Agent Theories to Agent Implementations (AT2AI-4), Vienna, Austria, 13-16 April, 2004.
- [17] V. Ermolayev, N. Keberle, S. Plaksin, O. Kononenko, V. Terziyan "Towards a framework for agent-enabled semantics web service composition". *International Journal of Web Services research*, 1(3), 63-87, 2004
- [18] [www.gria.org](http://www.gria.org) GRIA



