Towards an Abstract Service Architecture for Multi-Agent Systems

Jonathan Dale
Fujitsu Laboratories of America
jdale@fla.fujitsu.com

Margaret Lyell
MITRE Corporation
mlyell@mitre.org

Abstract
One of the common advantages of the agent paradigm is that it can deal with whatever technologies are appropriate to achieve their intended goals. In terms of interacting with *services*, this means being able to deal with the current set of Web Service technologies, as well as DAML-S, the GRID, etc. Each of these service technologies has different structures, formats, requirements and goals which can make it difficult to provide relationships between them. The work presented in this paper defines an abstract service architecture which is being developed as part of the FIPA software agent standard in which multiple service technologies can be expressed; agents can then use this architecture to provide points of commonality between different service technologies.

Keywords Web Services, DAML-S, Agents. FIPA, Service Architecture

1. Introduction
Over the past few years, legacy applications have become Web-enabled to allow for browser (human) access to data. A current trend is to wrapper legacy applications with Web Services technologies to allow for (machine) data exchange between legacy applications and software clients. This is one of the current technology iterations of making computer information automatically processable by software; there are others, such as the Semantic Web, the GRID, etc.

The software agent paradigm has been put forth as being appropriate for the software engineering of complex, loosely coupled systems [1]. As more application software is developed that utilizes software agents as components, it is likely that a need to expose various functional parts of such applications as services will arise. Support for such services requires the development of a communications framework consisting of, amongst other things, data formats, transport protocols, communication interactions, etc.

The Foundation for Intelligent Physical Agents (FIPA) is a standardisation effort for describing the framework of communication between agents to allow software to interoperate across the Internet. In the present set of FIPA specifications[1], the concept of a service is defined in the FIPA Agent Management Specification [2], primarily through a data structure type of entity known as the FIPA Service Description. However, general issues have arisen as to how services should be represented and how these services should be described, for example, complications occur when the service is composite. Also, the FIPA Service Description does not provide the scope for an adequate description of a service.

To help address these inadequacies and also to help provide a common framework in which all service models and service technologies can be represented, the FIPA Services Technical Committee is working towards the development of an abstract architecture for Services. An abstract architecture is on in which only the essential components are described and their relationships without any explicit use of or reference to a specific technology. Such an architecture allows the designers to focus on the overall design and interaction of the system without becoming involved in technical details. From the abstract service architecture, the FIPA Services Technical Committee will also produce a number of reifications, which are concrete realizations of the abstract architecture, initially for WSDL and DAML-S.

In this paper, we aim to describe the abstract service architecture that the FIPA Services TC has developed and show how it can be used to represent and express other service technologies. In section 2, we briefly discuss the current service manifestations of Web Services and DAML-S. This is done in order to provide background for Section 3, which discusses the current status of FIPA specifications with regard to agent-based services, some issues and relationships to the extant service approaches. A snapshot of the current work on the abstract service architecture is presented in Section 4. Future directions are discussed in the final section.

2. Current Service Paradigms
The *service* concept is germane to many current technologies, including Jini™, software agents, the Grid and the Semantic Web with DAML-S, as well as the area of contemporary Web Services with its representative technologies of SOAP, UDDI, WSDL, BPEL4WS. While each of these technologies has a distinct focal point, there is overlap in the application areas that they purport to serve. Furthermore, the continued evolution and maturation of these technologies may lead to convergence.

2.1 Web Services
The current instantiation of what is commonly known as Web Services incorporates the range of technologies necessary to provide for the discovery and invocation of services. The message format used in service invocation is described by the Simple Object Access protocol (SOAP) [3] and a description of the message interface for a particular service is given in its (publicly accessible) Web Service Description Language (WSDL) [4] interface file. The WSDL description contains the binding information for the particular service.

Services can be registered and discovered by using the Universal Description, Discovery, and Integration (UDDI) [5] registry. There is a connection between a service's description in the UDDI registry and its WSDL file(s) which are facilitated by specific data structures. Through the *businessService* data structure, a service is registered in the UDDI. Information on how/where the service is accessed is provided by a service's *bindingTemplate* and a service may have more than one *bindingTemplate*. The WSDL interface

[1] See http://www.fipa.org/

information for a service is associated with a *tModel* data structure in the UDDI registry. The *bindingTemplate* contains a reference to the *tModel*.

The UDDI registry also provides for service advertisement through the use of entries in a *categoryBag* data structure. The entries can be attribute/value pairs which can refer to industry standard information. However, the usage scenarios involving the UDDI registry do not focus on dynamic discovery of a service by a client. This is a critical aspect of service discovery in multi-agent systems where agents may need to find services dynamically and compare different service offering sets at different points in time.

Service composition is the process of building new services from the combination of existing services and is being addressed by the Business Process Execution Language for Web Services (BPEL4WS) [6] that represents the newest area of activity for Web Services. In BPEL4WS a composite service uses the WSDL files of its constituent services to provide a description of its message interface.

2.2 DAML-S

DAML-S is based on DAML+OIL and is an ontology for services [8]. In the DAML-S upper ontology, a *Service* must provide information on what it does, how it works and how it can be accessed. This is formalized through the classes of *ServiceProfile*, *ServiceModel* and *ServiceGrounding*, respectively. There are two cardinality constraints associated with DAML-S which is that there can be only one *ServiceModel* and there must be at least one *ServiceGrounding*.

Service usage is via a *ProcessModel* which describes the inner workings of a service, but there is no specified linkage between the *ServiceProfile* and the *ProcessModel*, so care must be taken to avoid inconsistencies. At present, there are three types of processes: *Atomic* (can be invoked and appears to be a single service to user), *Composite* (decomposable into other processes and services) and *Simple* (abstract). The *ProcessControl* ontology is needed in order to provide for execution and monitoring of service requests[2].

The DAML-S specification makes it clear that the language is intended to be used in conjunction with a suitable planning language, such as ConGolog [8]. Such a language would make use of the pre- and post-conditions specified in the service to assist in both automatically selecting services to help complete a plan and composing services (the post-conditions of one service would form the pre-conditions of another, compatible, service).

2.3 Services in FIPA

The FIPA Service Description in the FIPA Agent Management Specification [2] gives the current viewpoint of how services are viewed in the context of FIPA agents. What is implicit in this specification is that agents will be both the providers and users of agent-based services. Agent-based services are advertised with the Directory Facilitator (DF) which is a mandatory agent on an FIPA-compliant agent platform that provides service registry functions.

The FIPA Service Description data structure details the type of information that can be provided to advertise an agent-based service. It includes a description for owner, name and type of the service as well as the interaction protocols, ontologies and agent content languages that the agent offering the service can utilize. Finally, a list of properties in the form of attribute/value pairs describes service-specific information.

There are shortcomings with the current FIPA service description. The properties list is the structure that hosts the description of what a service offers and the lack of structure in this list makes it impossible to capture hierarchical relationships that could be required for proper definition. For example, there is no support for hosting a description of where to access additional descriptive information regarding the service and there is a lack of grounding facilities for multiple service technologies. This leads to questions such as the following which are not currently addressed by FIPA:

- How can a FIPA service description support a description that would be appropriate for a service that is to be advertised in both the FIPA Directory Facilitator and a UDDI registry?
- How can a service description be expressed both in DAML-S and as a FIPA service description?

2.4 Analysis

With regard to Web Services, DAML-S provides a semantically higher level and richer way of describing services than appears in either WSDL or in the descriptions in a UDDI registry. The WSDL interface descriptions correspond to a description of the message needed to invoke a service which is conceptually closer to the *ServiceGrounding* in DAML-S. The *ServiceProfile* of DAML-S provides a semantically meaningful description of services, as compared with that put forth by the data structures in the UDDI registry. However, DAML-S and related work does not provide a registry specification for hosting DAML-S service information. The DAML-S *ServiceModel* and *ProcessModel* are related to the efforts of BPEL4WS in the world of Web Services since the use of component services in BPEL4WS is described by WSDL type documents.

Within FIPA, it is an agent that offers the agent-based service. At present, the scope of Service Description registration is on the agent's home platform and on any other federated FIPA-compliant platform. This is a more restricted forum than that proposed by either the Web Services or Semantic Web (DAML-S) visions. However, it is recognized that early users of Web Services might be Intranets within organizational units. This is consistent with the scope of agent-based services on agent platforms.

Reiterating, it is an agent that offers the agent-based service. The service is not conflated with the agent's identity. A particular agent could offer multiple services. Communication with a FIPA-compliant software agent is via a FIPA Agent Communication Language (FIPA ACL) message, with content that is expressed in a suitable agent content language. Sufficiently expressive agent content languages enable content that can be reasoned over, with ontological support. This is qualitatively quite different from the specification of an interface in WSDL containing the method name and parameters necessary to invoke a service. Should all users of an agent-based service required to be agents?

[2] At the time of writing, the development of the *ProcessModel* in DAML-S is not yet complete.

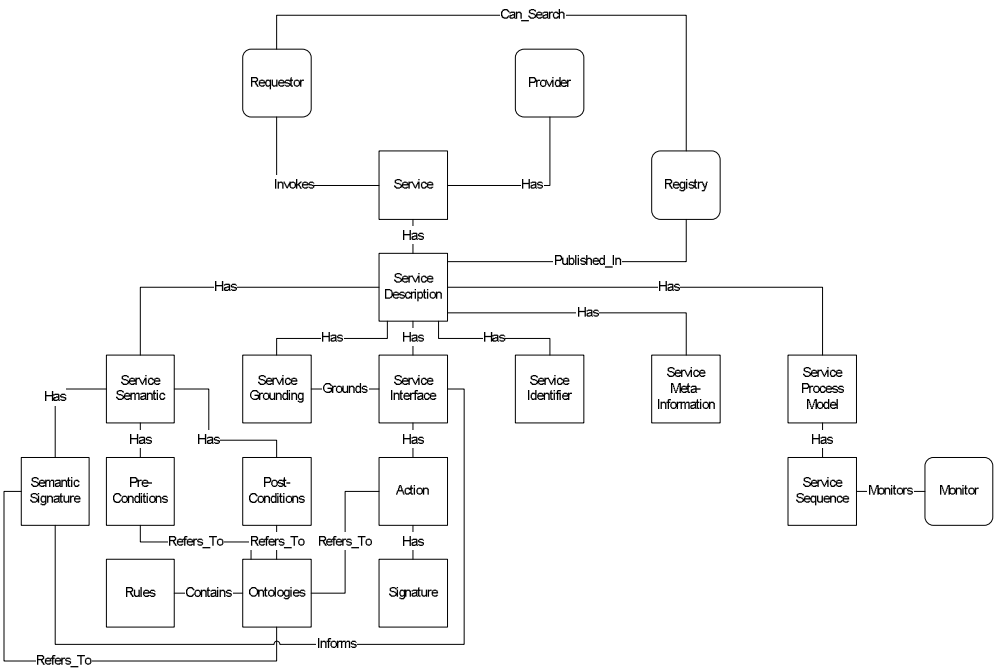

Figure 1: Abstract Service Architecture

FIPA agents could utilize lessons learned in the use of DAML-S, in particular, with regard to reasoning in support of service selection. Also, the organization of an evolved FIPA Service Description can benefit by consideration of the DAML-S effort.

With regard to the service grounding and actual service utilization, it is Web Services that are in the forefront. The WSDL service description distinguishes between the interface description of a service, and the implementation, or grounding. If an agent-based service is not going to be accessed simply via an ACL message with the appropriate content (expressed in an agent language), then provision must be made for a description of the service grounding.

3. Abstract Service Architecture

The FIPA Abstract Service Architecture is currently under development and the initial efforts which were made at the Palermo, Italy meeting in February, 2003 have been aimed at characterizing the service components and its relationships.

3.1 Structure

During this initial work, we identified the following abstract components (see Figure 1) of a *Service Description*:

- *Service Interface*. This is the interface of the service defined in terms of *Actions* each with a distinct *Signature.*
- *Service Identifier*. This is a globally unique identifier for a particular service instance.
- *Service Meta-Information*. This is a description of the service, such as its application domain, owner, etc.
- *Service Grounding*. This is a set of groundings of the service for various service technologies.
- *Service Process Model*. This is a description of the process steps that the service goes through to achieve its actions; there is one process model for each *Action.*
- *Service Semantic*. This is a semantic description of the actions of the service that other software can use in trying to determine the function and capabilities of the service.

Also, we have identified a number of roles in the Abstract Service Architecture which can manipulate services:

- *Provider*. The provider of a service or set of services; each service offering has a *service description* which can be published in a *registry*.
- *Requestor*. The user of a service; typically invoked by querying a registry and then analyzing a service's *service description.*
- *Registry*. A directory service where *service descriptions* of *providers* can be published and subsequently queried by *requestors*.
- *Monitor*. A control function which can monitor the status of a service invocation.

3.2 Operation

A *Service Description* is logically divided into two categories, that dealing with semantic information on the meaning of the service and that dealing with the service grounding. By analyzing this information, a user or software is then able to obtain information on why and how the service should be used.

A *Service Description* has a *Service Semantic* that comprises a *Semantic Signature* and *Pre-* and *Post-Conditions*; these allow for more advanced semantic interpretation and reasoning about the *Service* than its *Service Interface* alone. *Ontologies* assist in the

interpretation of the *Service*, since the *Pre-* and *Post-Conditions* of the *Service Semantic* as well as the *Action* of the *Service Interface* refer to them. *Rules*, associated with the *Ontologies*, can give additional information on the context in which the ontological information is appropriate. For example, a withdrawal from a bank account can occur only if the fund balance exceeds the withdrawal amount.

Services can be composed to provide more complex services. An agent acting in the composer role can inspect registered *Services*, which implies that the composer agent is inspecting services that involve the same (or overlapping) ontologies. For example, consider two services, A and B. In order to generate a third service, C, the post conditions of A must be acceptable as the pre-conditions of B. This is assessed using the information associated with the *Service Semantic* of A and B. However, due to the complexity of this process, the composition operation might not be performed at run-time (and may be beyond the capabilities of current software engineering practices). Rather, agents who fulfilled the composer role would traverse the registered services, inspecting the *Service Semantic* and *Service Interface* information, generating potentially useful composed services, perhaps in collaboration with user intervention. Thus, the composer agents can help to automate the process of leveraging the already extant service assets of the enterprise.

The orchestrator role for an agent, which involves the execution of composed services, has not yet been introduced into the Abstract Service Architecture but will be the subject of future efforts. Reflection upon the orchestrator role is likely to yield insight into the distinctions between the concepts of agents and services.

4. Conclusions and Future Directions

Extensions to the current state of the FIPA Abstract Service Architecture that will incorporate service composition and orchestration are planned for the near term. A reworking of the FIPA Service Description that provides at least a minimal supporting structure to convey grounding information is necessary. Actual service groundings will not be addressed until after these conceptual tasks are completed.

Two sets of specifications are planned:

- Abstract Service Specification. This is a more complete description of the Abstract Service Architecture and how it is intended to function. It will also provide groundings for service technologies such as DAML-S and Web Services as exemplars of the system.
- FIPA Service Description Specification. This is a set of extensions to the current FIPA Agent Management specification that addresses some of the existing criticisms mentioned in section 2.3.

This activity of the FIPA Service TC, an eighteen months effort, started in February, 2003 and is planned to be completed by April, 2004.

ACKNOWLEDGEMENTS

The authors would like to thank all of the participants of the FIPA Services TC that helped to refine the initial Abstract Service Architecture proposal at the Palermo, Italy meeting during February, 2003.

REFERENCES

[1] *An Agent-based Approach for Building Complex Software Systems*, Jennings, N. In: Communications of the ACM, 44(4), pp 35-41, 2001.

[2] *FIPA Agent Management Specification [FIPA00023]*. Foundation for Intelligent Physical Agents, 2002. `http://www.fipa.org/`

[3] *SOAP Specification*. World Wide Web Consortium, 2002. `http://www.w3.org/TR/soap12-part1/`

[4] *WSDL Specification*. World Wide Web Consortium, 2002. `http://www.w3.org/TR/wsdl/`

[5] *UDDI Specification*, OASIS, 2002. `http://www.uddi.org/specification.html`

[6] *BPEL4WS specification*. IBM, 2002. `http://www.ibm.com/developerworks/library/ws-bpel/`

[7] *DAML-S: Semantic Markup for Web Services*. The DAML Services Coalition, 2002. `http://www.daml.org/services/daml-s/0.7/`

[8] *ConGolog, a Concurrent Programming Language Based on the Situation Calculus*, De Giacomo, G., Lesperance, Y. and Levesque, H. In: Artificial Intelligence, 1-2(121), pages 109-169, 2000. `http://citeseer.nj.nec.com/degiacomo00congolog.html`