# Standardizing Agent Interoperability: The FIPA Approach

Stefan Poslad[1] and Patricia Charlton[2]

[1] Department of Electronic Engineering, Queen Mary, University of London,
Mile End Road, London, E1 4NS.
stefan.poslad@elec.qmul.ac.uk
[2] Centre de Recherche de Motorola-Paris
Espace Technologique – Commune de Saint Aubin
91193 Gif-sur-Yvette, France.
patricia.charlton@crm.mot.com

**Abstract.** A prolific number of different Multi-Agent Systems (MAS) and associated applications have been developed in numerous research institutes and industrial laboratories world-wide. Perhaps the most important barrier to MAS making a successful transition from this research environment towards widespread adoption for consumer products and businesses, is the lack of interoperability between heterogeneous MA Systems. In 1996, the **F**oundation for **I**ntelligent **P**hysical **A**gents (FIPA) was formed to provide a forum for developing specifications for agent systems. Since its formation, FIPA has increasingly focussed more on standardizing (multi-agent system) agent interoperability. As a result, it is often said that FIPA really stands for the **F**oundation for **I**ntero**P**erable **A**gents. In this article, we discuss both technical and scientific issues in defining standards for interoperability between agents in different MA systems with a particular focus on the FIPA agent interoperability standards.

## 1 Introduction

In 1996, FIPA, The Foundation for Intelligent Physical Agents [1], a forum of international companies with a strong focus in the telecommunication industry, was formed to promote the uptake of software agents in businesses at large. It was originally intended that specifications and standards, encompassing both hardware (physical) agents such as robots and software agents, would be developed hence the use of the term physical in the FIPA full name. However, as FIPA progressed, the interest of the forum focused increasingly on software rather than hardware agents. A second key focus was on specifying communication and interoperability between agents rather than specifying how agents processed and reasoned about the information they received. That is, FIPA concentrated on standardising 'external intelligence' or rich interoperability rather than on 'internal intelligence' or reasoning. FIPA's official mission statement is: "The promotion of technologies and interoperability specifications that facilitate the end-to-end interworking of intelligent

M. Luck et al. (Eds.): ACAI 2001, LNAI 2086, pp. 98-117, 2001.

agent systems in modern commercial and industrial settings." Hence, it is said that perhaps FIPA more accurately stands for "*F*oundation for *I*ntero*P*erable *A*gents" or "Foundation for Interoperable Peer-to-peer Agents.

FIPA produced its first set of seven specifications in 1997. This set of specifications included: an agent architecture referred to as the FIPA 1997 agent platform; an agent communication language and some applications such as travel assistance, network management, audio-visual entertainment and personal assistance In subsequent years, further specifications were added, the architecture and agent communication language have been refined, the specification process has been formalized and there is more focus on abstractions and instantiations of the abstractions for a heterogeneous world.

The remainder of the article is structured as follows. We first (in the following two sections, 2 and 3) try to give some insight into why and how FIPA has developed specifications in a particular way. Then we summarise the specifications (section 4). Then, in section 5, issues governing the implementation and use of the specifications are considered. Finally, a conclusion about FIPA's work is given.

## 2 Modelling a Richer Type of (Agent-Based) Interoperability

Multi-agent architectures are distributed systems in which each agent is: able to act autonomously, able to reason and make decisions about the environment in which it is embedded, and able to interact with other agents. The distributed and autonomous nature of agent systems potentially supports flexibility and robustness in system operation and organization, particularly when these are coupled with dynamic system behaviour. Highlighting "agent systems" as autonomous helps to highlight the usefulness of agents in solving complex real world problems.

As the area covering agents and agent technology is a heterogeneous body of research and development, there are several dimensions in which one can characterise agent software. Frankin and Graesser [2] identify the following dimensions to characterise agent software: reactivity, autonomy, proactivity, responsibility, continuity, interactivity, adaptability, rationally, cooperativity and robustness. The above properties are considered generally useful for an agent. However, it should be noted that conditions for agent-hood are not necessarily environmentally determined. The environment will determine which properties favour success and which cause failure. Thus, there are of course other properties that are not critical to a general notion of agent-hood, but that can be used as a basis for the classification of agents. One example is agent mobility: an agent that is able to transport itself from one site to another across a network. Mobility however does become a necessary condition of agent-hood in certain environments, those where hard real-time constraints exist at remote servers, where large volumes of data must be processed, where processing resources of agent servers are over-utilised, and where the communication channel is costly or hostile.

For the purposes of our discussion of the FIPA specifications, we focus primarily on the autonomy and interactivity dimensions of agents. In order to usefully exploit

both the autonomy and interactivity, engineering solutions using agents, the partitioning of the domain or problem by explicitly abstracting certain characteristics of the domain and associating these characteristics with an agent's role is required.

At its core an organisational approach [3], rather than a top-down or bottom-up approach. There are several organisational-related aspects that are important factors influencing the identification and characterisation of agents. Some of these aspects are:

- an organisation can be divided into sub-groups having a cohesion that makes it natural to associate an agent with a specific sub-group.
- within a group, certain services should be provided, and services that to a large degree share domain knowledge can be realised by a single agent.
- a larger group might be partitioned into separate responsibilities, and this can cause separation of services into different agents for each responsibility.
- a group can be governed by a set of rules or policies (constraints) identifying what are legal states and behaviours, and agents should exhibit a behaviour that conforms to these rules.

Hence, to solve distributed complex problems using multiple, autonomous, communicative agents, we partition the problem into sub-problems and map these to one or more organisations of agent groups with a common behaviour and then to agents with individual roles that co-operate with other agents in the same group or in different groups.

There are two main ways that multi-agent systems can be modelled or expressed: they can be designed using descriptive frameworks, accompanied by some prescriptive element in order to constrain the behaviour or they can be modelled using a formal logical framework, whose behaviour is well defined and can be verified. The FIPA specifications contain elements of both these approaches. It is important to consider these because the definition of the model will affect how different implementations of the model can interoperate. These two approaches are now examined in more detail in turn.

### 2.1 FIPA Agent Specifications as Descriptive Models

There are many descriptive frameworks that describe and classify agent behaviour, for example, [2], [4] and [5]. Descriptive system frameworks can contain several different complementary views of the same system: a functional or behavioural model, a data or information model, an organisational model and an interaction or operational model.

The functional view or model defines the properties and behaviours of agents using informal descriptions. The functional view (and the operational and organisational view) can also define prescriptive elements: rules such as pre and post-conditions that determine when the function can and cannot operate. An interaction or operational model defines the sequences of functions that define the (normal) operation of the system. The data or informational model expresses the structure of the information for exchange and for persistence. The organisational model defines how parts of an agent are structured, how different types of agent are related and how instances of agents

can form (organisational) groups that offer higher-level abstractions. These models can have different levels of granularity.

The FIPA specifications use all of these four types of model to define multi-agent systems. A top-level organisational model for FIPA agents is given in Figure 1. This is slightly adapted from that in the FIPA Abstract Architecture specification [6].

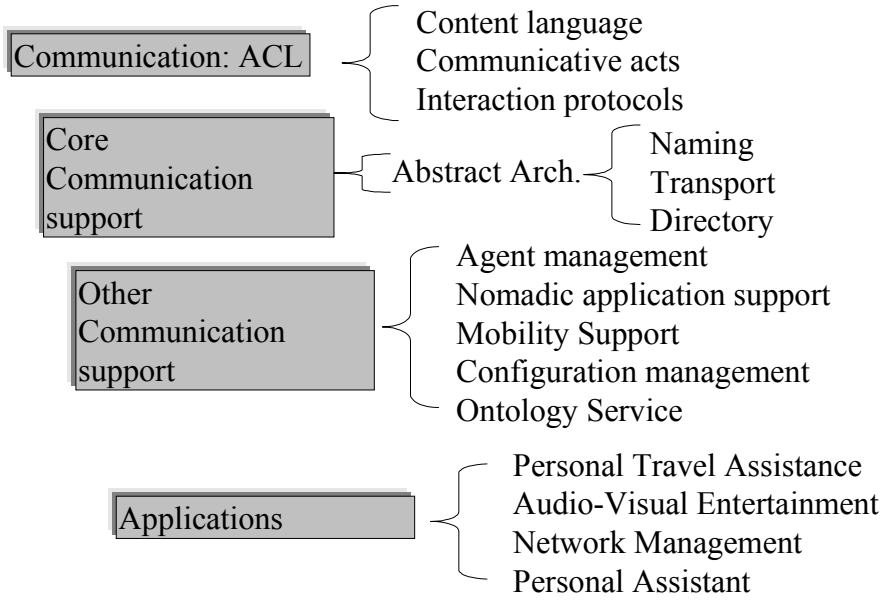


**Fig. 1.** A high-level organisational model of FIPA multi-agent systems

The functional model describes the behaviour of an agent component, for example, the following extract describes the function of the directory service as described in [6]. "The basic role of the directory-service function is to provide a location where agents register directory-entries. Other agents can search the directory-entries to find agents with which they wish to interact. The directory-entry is a key-value-tuple consisting of at least the following two key-value-pairs…In addition the directory-entry may contain other descriptive attributes, such as the services offered by the agent, cost associated with using the agent, restrictions on using the agent, etc…"

An example of a FIPA informational model is given in Table 1 and is taken from the FIPA transport specification [7]. This defines the header or wrapper that is added to ACL messages when they are transported between agents.

An example of a FIPA operational model is given in Figure 2, and is taken from the FIPA interaction protocol specification such as the FIPA-Request-Protocol [8]. This specifies a dialogue for one agent to request another to perform some action, and the receiving agent to perform the action or, to reply in some way that it cannot. The dialogue is represented in an agent extension of UML (Unified Modelling Language) called AUML or Agent UML [9]. This extension adds support for concurrent threads

of interaction and the notion of an agent role. Pitt et al [10] have proposed a formal semantics for AUML.

| **Frame Ontology** | envelope FIPA-Agent-Management | | |
|---|---|---|---|
| **Parameter** | **Description** | **Presence** | **Type** |
| to | This contains the names of the primary recipients of the message. | Mandatory | Sequence of agent-identifier |
| from | This is the name of the agent who actually sent the message. | Mandatory | agent-identifier |
| comments | This is a comment in the message envelope. | Optional | String |
| acl-representation | This is the name of the syntax representation of the message body. | Mandatory | String |

**Table 1.** A FIPA informational model: fragment of the header for ACL messages.

Inherent in dialogues are constraints on the order of the messages in the dialogue, the messages can only occur in a certain order during normal interoperation between the sender and receiver. This is an example of a prescription. Prescription constrains the operation of the system to better support system management and interoperability. Prescription expresses the conditions under which elements can be normally be used. If these conditions are false, then either errors must be raised and handled or the system operation is suspended until the conditions are true again. Embedded in the specifications are many prescriptive statements or rules that to constrain elements of behaviour of FIPA agent systems.

For example, here are examples of prescriptive constraints from the FIPA agent management specification [12]:

- "An agent must have at least one owner .."
- "A Directory Facilitator (DF) is a mandatory component of the AP (Agent Platform) .."

The following rules are adopted to select the appropriate communicative act that will be returned when a management action causes an exception:

- "If the communicative act is not understood by the receiving agent, then the replied communicative act is not-understood."
- "If the requested action is not supported by the receiving agent, then the communicative act is refuse."

To summarise, the FIPA specifications are expressed using different types of descriptive models with various (informal) prescriptive rules to constrain the operation, behaviour and functions of FIPA agents.

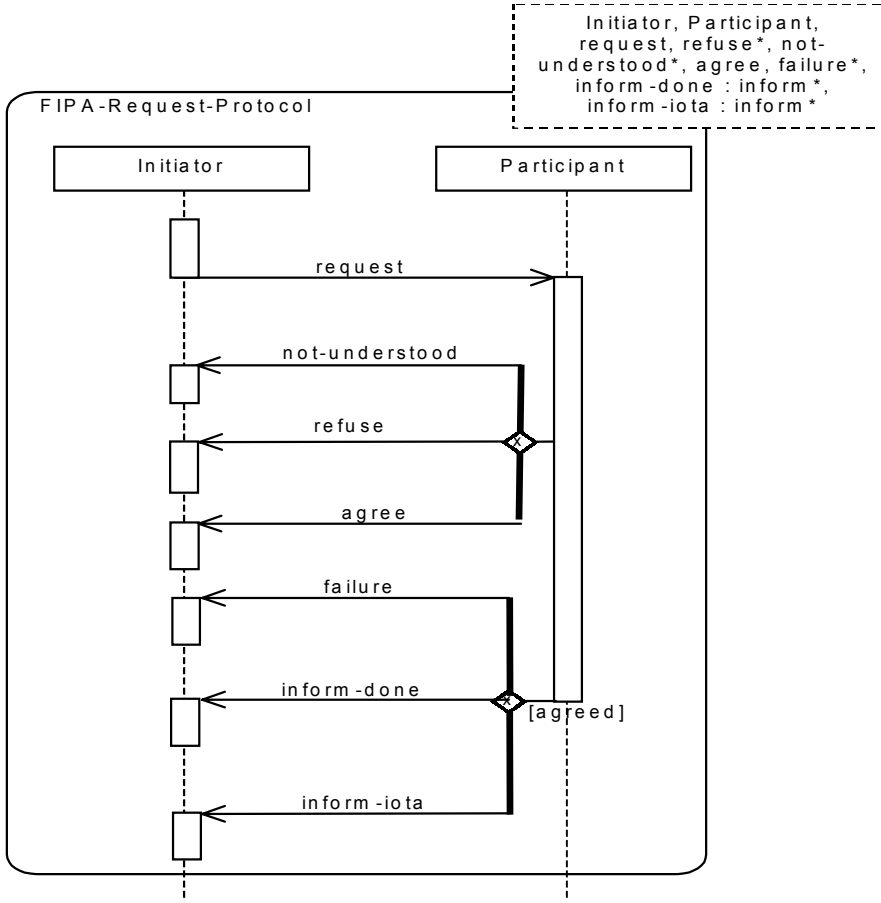


**Fig. 2.** A FIPA interaction model: the FIPA-Request Interaction Protocol

## 2.2 FIPA Agent Specifications as Formal Models

An alternative approach, to employing some kind of descriptive framework to define agent behaviour and in particular agent communication, is to use a more formal model to govern the agent behaviour. For example, the semantics of FIPA agent communication is (partially) underpinned by a formal logic. The FIPA ACL has formal semantics that are defined using a multi-modal (BDI-type) logic called Semantic Language or SL. The semantic model maps each agent message type (speech act) to an SL-formula that defines constraints that the sender must satisfy called the feasibility conditions. FIPA also maps each message to an SL formula that defines the rational effect of the action.

Putting aside theoretical issues such as whether or not such a formal model is well-defined or well-grounded, we could compute when each message is sent that it obeys the feasibility and rational effect conditions and hence we could verify that the message conforms to the FIPA formal-logic specification. This allows standard axiomatic proofs to determine and ascertain the behaviour of each aspect of the system. For example, if we have axiomatic proofs to describe the transmission of messages between a sender and a receiver and for describing the effect these messages cause at the receiver and the sender then we can ascertain when we have the correct behaviour for successfully transmitting a message. This would theoretically make maintaining and explaining the causes of interoperability easier.

### 2.3 Descriptive vs. Formal Models

When considering the form of specifications for supporting agent interoperability, there are persuasive arguments for using descriptive approaches, for using formalisms or for using a hybrid approach.

There are many perceived problems with the current formalism that underpins the FIPA ACL - SL. Firstly, the formalisms within the FIPA specifications are only used to model (simpler?) parts, such as the state before and after sending a particular speech act, of agent communication. For example, it does not explain how dialogues work, how agent systems can be managed and maintained or what do if partial failures occur in a distributed system. Secondly, various researchers have argued that the particular semantic logic used by FIPA is too limited to support (the various kinds of) agent interoperability. For example, Wooldridge [13] argues that the SL formalism does not lead to proofs that can be easily automated or computed – SL is "ungrounded". It is also generally accepted that intentional semantics are problematic within a distributed environment [14] hence the SL semantics for the rational effect, at the receiver, are not generally observed by many FIPA agents in practice. It is also proposed that different semantics are needed to better support agreements and observable commitments [10].

Thirdly, many particular logics may not be expressive enough, in practice, to capture the complexity of the real world; to be computationally tractable in a timely fashion and to be verifiable in a non-deterministic world or to function in a dynamic, extensible and open system environment.

There are limitations to a purely descriptive approach to specification even if that approach is complemented with prescriptive elements. It is simply not enough for specifications to gain widespread support and become a de facto standard. To gain, and maintain acceptance, particularly for sensitive applications such as electronic commerce, it must be possible to determine whether or not any system that claims to conforms to an ACL standard actually does so. An ACL standard is verifiable if it enjoys this property [13]. Traditionally, the way distributed systems are modelled, using descriptive systems, to show conformance to a standard that is specified in some descriptive framework, is to show that the actual output of the system, given some predefined input, equals the expected output. The weakness of this approach is that conformance, at points other than the defined conformance points, is undefined. A second complimentary approach is to define pre and post-conditions for the computation to determine when a function should be called and what the result should

be after the function has been triggered. This enables the (correct) behaviour of the function to be verified (according to the computation conditions).

Both descriptive and formal approaches to specifications have strengths and weaknesses. If FIPA agents are to be used in domains such as eCommerce then verifiable models of operation and for interoperation are essential. The use of both descriptive models with prescriptive elements and formal, computationally grounded models of operation are important.

## 3 Standards for Multi-agent System Interoperability

In a heterogeneous world, concurrent distributed development has lead to many types of multi-agent systems that are islands of functionality – agents on different types of platform are unable to interoperate with each other. Agents from different vendors are likely to use different types of messages and message formats and the meaning and interpretation of the content is likely to differ. The driving force for interoperability is partly the customer who strives for simplicity and universality when accessing multiple services, and partly producers who often need to act in unison to obtain a critical mass for a sustainable customer-base.

Early adopters, who produce new technology and services, tend to be wary where there is no commonly agreed standard for interoperability and suspicious of standards that lack the support of a large consortium of companies. The standardization process helps shift the emphasis from the development of the infrastructure to the use of the infrastructure.

The main driving force for de facto standards from multiple-vendor forums is to seed the market and attain a critical mass of customers, applications and products in the medium and long-term. The seeding of the market is enhanced when the standards are publicly available - they are easier to be taken up by non-members. In standardizing, companies potentially give up a competitive edge in the short-term goal but regain it in the medium to long term by producing products that add value and enhance the standard in a much larger market.

Standards need to be developed at the right time. The development of new products and markets seems to follow a double peak of activity with a dip in between. The optimum time to standardize is towards the end of the first (research) peak or in the dip between the peaks, before the second (development) peak occurs. Standardizing too early before the research is finished can lead to immature, bad standards whereas standardizing too late, once companies have made significant investments, can mean the standards are ignored. We regard MAS to be somewhere in the dip between the peaks at this stage.

### 3.1 Agent-Based Interoperability

The electronic service space for e-commerce current and future direction requires advancement in distributed infrastructure. Key to this is the interoperability of intelligent distributed systems. This paper provides an initial analysis of the

requirements for high-level interoperability of services in an intelligent yet distributed and dynamic manner.

Interoperability is at many levels and this approach represents interoperability as a high-level. The importance of this is to recognize all levels and the impact of each level on reaching semantic integration, which is both flexible and extendable. Often, the focus on just one level (syntax) and which limits the possibilities of providing the type of infrastructure necessary for dynamic and new e-services.

In particular we wanted to bring out the following important features:

- General components can be designed to map to interoperability needs,
- Openness, which can be achieved through providing structured and explicit interfaces of interoperability i.e. a standard is essential. Although business models may not want complete openness for economic reasons it is necessary to have a certain degree to support services on the advancing internet in any meaningful way
- Frequently, interoperability in agent systems is only considered at some communication level. This is not enough for the future of service support but is an essential starting point.
- The meaning of an action of communicating is at a different level of interoperability – it is more than just sending a message. The content of the message requires an ontological definition. However, a ontological definition can vary to such an extent that semantic interoperability can not be achieved and certainly service integration in any open sense is lost. Ontologies are still defined and abstracted by hand. The automation of this process is still a main issue but aspects are being resolved – see [1] and [15]

Although standards (for specifications) can underpin interoperability, well-defined specifications in themselves are not sufficient for interoperability - general characteristics of the specifications such as scope, extensibility and form will impact the interoperability and these need to be assessed. As the relationship between the form of the specification and interoperability has already been addressed in Section 2, the scope and extensibility aspects will be debated in the remainder of this section. A discussion of the content of the specifications will be deferred until Section 4.

### 3.2 The Scope of Standards

The content and boundary of agent specifications is contentious for several reasons: because the type and variety of agents supported and the type of interfaces to the supporting computation and communication infrastructure for agents, are debatable. Here an overview of the scope and content of the FIPA agent specifications will be debated. First some general statements are made then some of detail of FIPA's approach will be explained in each case.

*FIPA focuses on specifying external communication between agents rather than the (internal) processing of the communication at the receiver.* The organisational schematic model in Figure 1 illustrates this: FIPA agent specifications cover four main areas: agent communication, core support for agent communication, other support for communication and application domains. FIPA has not standardised the internal processing of communication because it is too contentious, at least at this

time. instance, it would be difficult to define a sufficiently generalised, and hence standardised, inference engine that could infer what to do when particular types of agent messages are received.

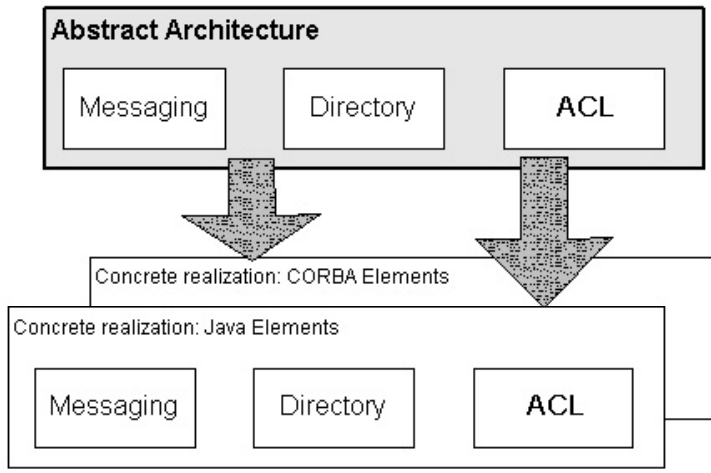


**Fig. 3.** The abstract architecture and its realisation

This introduces a second important point that *FIPA specifications, foremost, attempt to cover generalisations and high-level neutral abstractions*. The core FIPA specifications are neutral with respect to:

- a specific service or end-user application;
- a particular supporting software and hardware infrastructure and implementation such as a computer language.

For example the FIPA Abstract Architecture [6] defines a high-level organisational model for agent communication and core support (minimum support required) for communication such as a directory service, message transport service and namespace. The abstract architecture is neutral with respect to any particular directory service or the use of a particular network protocol for message transport. The abstract architecture itself cannot be directly implemented, but instead forms the basis for the development of concrete architectural specifications (see Figure 3). Concrete implementations can implement all or part of the specification.

Thirdly, specifications are often dependent on other horizontal layers, e.g. they may use an existing software infrastructure. *A key issue is how much leverage to make from existing (non-agent) technology and how to link the agent parts to an existing infrastructure*. The scope of MA specifications generally includes the interpretation and handling of ACL (Agent Communication Language) messages, facilitator agents, and the use of, but not the actual specification of an existing software infrastructure such as message transport protocols, and message persistence schema, to underpin agent communication.

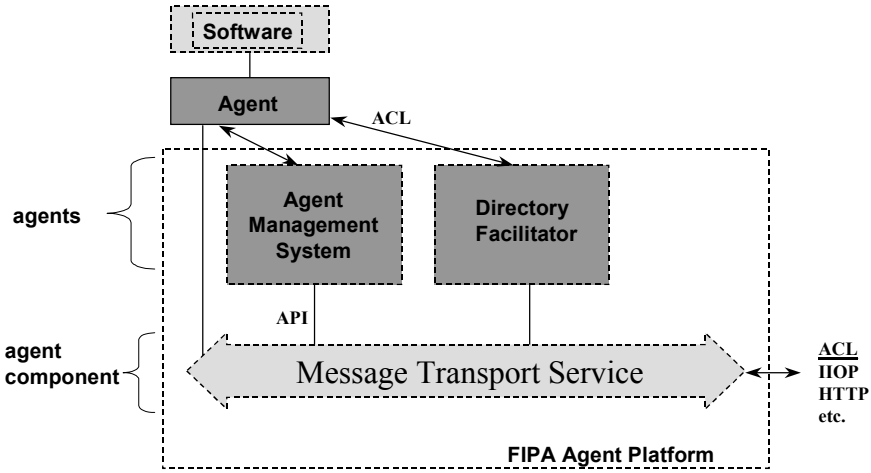


**Fig. 4.** The FIPA agent platform

Fourthly, different levels of granularities need to be modelled: we need to distinguish between agents and the level of granularity below that - the components that form the agent but, which in isolation are not an agent. *A related issue is determining what features should be modelled as agents and which should be modelled as non-agent parts*.

The following is an example of something that was first modelled as an agent but was later modelled as a non-agent part. The initial FIPA agent architecture was in essence taken to correspond to the agent platform defined in the agent management specification [12]. In the early versions of the specification (1997-1998), the message transport specification was modelled as an agent. At first sight this seemed to offer great flexibility: because the transport agent is an agent, all agents can interact with it in a standard way and with a potentially very flexible and semantically rich ACL messages. But there is a downside – efficiency. To transfer a single message between agents always required sending at least two messages, one to ask the agent transport agent to send a message, the other for the agent transport to actually send the message. Hence, in later versions of the Agent Management Specification [12], the message transport is a non-agent service that can be invoked via some Application Programmer's Interface rather than via an ACL message (agent interface), see Figure 4.

It is also worth mentioning that the FIPA Abstract Architecture [6] requires no single generic service such as naming and the directory service to be an agent (but they may be an agent), whereas the FIPA agent management specification mandates the use of a combined name and agent life-cycle management service (called the Agent Management Service or AMS) and the directory service (called the Directory Facilitator or DF) to be agents. Although, these appear to be mutually exclusive, they are not necessarily so. If agent management is present (it is optional according to the Abstract Architecture Specification) and it adheres to the agent management specification, then it requires agent based name and agent based directory services. If both the name service and directory service (and transport service) adhere to the

properties and interfaces defined in the Abstract Architecture then the particular use of the agent management specification can also adhere to the FIPA Abstract Architecture specification.

### 3.3 Extensibility, Openness, Interoperability and Boot-Strapping

It is highly desirable that agent specifications are sufficiently extensible and open to work in a heterogeneous and changing world. This extensibility and openness is desirable at two distinct levels of granularity: at the agent level and at the agent component level (the components that underpin the agent).

At the agent-level, if agent systems are to scale up in the market-place, leading to mass-market penetration, then openness with respect to multiple vendors being free to add new agents and aggregate agents within a market-place, collaboratively, competitively and dynamically is highly attractive [17,18].

At the agent component level, it is desirable that the interface between the agent component and the agent does not bind the agent to a single particular instance of the agent component. For example, let's consider the agent transport as an agent component. In early versions of the Agent Transport Specification (in 1997 to 1998, it was actually part of the Agent Management Specification), FIPA specified the use of a single so called base-line message transport, the Object Management Group IIOP transport, which was ideal for use in low volume transaction, wire-line, private networks (without firewalls). However when FIPA agents were being considered for use via firewalls, for high-transaction processing and for wireless environments, the IIOP transport was considered to be less than ideal. It then became clear that agent component interfaces needed to neutral and abstract, e.g., the Agent transport specification (and the Abstract Architecture specification) can support multiple message transport protocols.

This openness to support bind agents to multiple instances of non-agent components such as multiple agent transport protocols introduces new interoperability problems. How can we bootstrap agent systems that are potentially using multiple transports? The Abstract architecture says that gateways will be needed to interlink different types of non-agent component but it does not deal in detail with bootstrapping. The specification of a base-line protocol, or in general the specification of a mandatory type of non-agent component that must always be there is useful because it means for the case of the transport protocol, the IIOP base-line could be used to negotiate the use of a more optimal transport protocol during the session. However, the use of a baseline even for bootstrapping may be unsuitable.

## 4 FIPA Standards: Engineering Issues

### 4.1 Overview of the Specifications

We have classified and packaged the agent specifications into four groups (Figure 1):

1. FIPA agent communication language (ACL) specifications
2. FIPA core communication support services

3. Other FIPA Communication support services
4. FIPA Application services.

The two sets of specifications that fundamentally define FIPA agent systems are groups 1 and 2 (2 corresponds to the FIPA Abstract Architecture [6]). In practice all of the currently open source implementations also adhere to two specifications in group 3: the Agent Management Specification [12] and the FIPA Message Transport Specification [7].

The specifications in groups 1 and 3 separate an abstract interface from a particular concrete realisation to promote specification extension and maintenance. For example, if another content language or message transport is added, the abstract interface specification part remains unchanged and a new concrete part is added.

Groups 1 and 3 are considered in more detail below. The second group has already been partially discussed in the scope section earlier. The fourth group of application services (see also Figure 4) provides an excellent starting point to understanding how the FIPA approach works in practice by using a single worked example for illustration within a single (application) specification document.

### 4.2 FIPA Agent Communication Language Specifications

The FIPA agent communication specifications also referred to as the the *Agent Communication Language* (ACL), are based on speech act theory [18] consist of:

- a fixed core set of about speech acts or communicate act messages
- a fixed core set of interaction protocols (the FIPA specifications refer to these sometimes as just "protocols")
- an extensible set of content languages.

N.B. The use of the term Agent Communication Language is overloaded: sometimes it is used to refer just to the speech acts, sometimes to the combination of the speech acts, interaction protocols and sometimes to the encoding of this combination for message transfer.

The set of speech acts form the basic set of message types exchanged between agents. Often messages are more usefully modelled within the context of a dialogue, an exchange of several messages between the sender and receiver. For example, client-server systems frequently exchange messages within a request-reply dialogue. The FIPA interaction protocols support richer dialogues and include task allocation (contract net), negotiation (auctions) and active directories (subscription).

The content is the part of a message that represents the domain dependent component of the communication. A content language is used to encode the *content* of a message. FIPA does not provide a single mandatory content language, but a set of reference content languages for a heterogeneous world such as Semantic Language, Choice Constraint Language (CCL), Knowledge Interchange Format (KIF) and W3C's Resource Description Framework (RDF). It is implied that the terms used in the content language are defined in an explicit ontology (at a simple level, the ontology can be considered as a domain specific dictionary of terms and the definition of any relationships between those terms) referenced in the ACL.

Finally, two levels of communicative syntactical wrapper are used when transmitting the message in practice, the first of these is sometimes (also) referred to as the ACL message structure [19], this defines the sender, receiver, the

communicative act, interaction protocol, ontology, content language and the content for the current message. A second or outer syntactical wrapper is used to wrap this message for message transport.

### 4.3 FIPA Specification Development and Maintenance

Specification and standard development requires a structured approach. A more structured process and life-cycle for specification development and maintenance was introduced in 2000 in part to address the following issues, pre-2000:

- it was not clear prior to 2000 how mature a specification is: whether or not it had just been released, whether it was currently being evaluated or whether or not the specifications had been road-tested;
- the design decisions for modifying the specifications were not captured and indexed.

FIPA specifications are given a status which defines their position in FIPA's specification life cycle (Figure 5). A specification begins life as a Preliminary status specification by a Technical Committee when it is working on a draft. When this TC decides that the draft is ready for implementation, it may be advanced to Experimental status and is frozen for a period of two years. After successful implementations and field trials of this specification, it may be advanced to Standard status, which is considered to be a stable and mature specification ratified by FIPA and implemented by organisations and companies. If a specification becomes unnecessary or is superseded by another specifications, then it is Deprecated for a period to six months to allow developers to change their implementations accordingly. After this period, the specification is made Obsolete - this means that it is no longer a supported specification of FIPA.

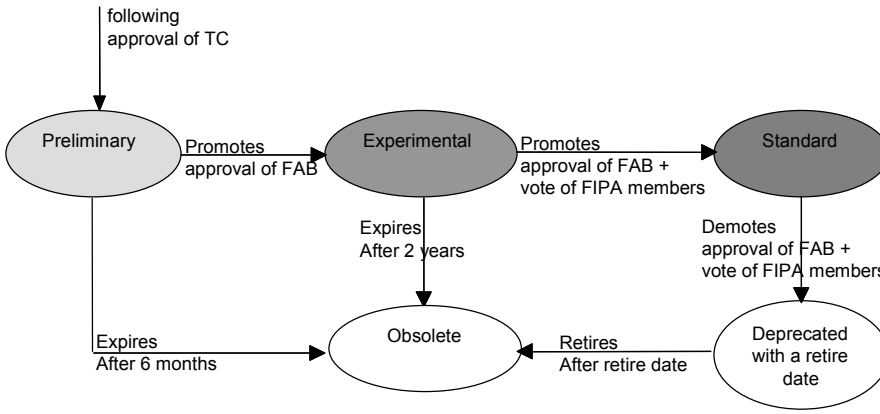


**Fig. 5.** Life-cycle of FIPA specifications. Specifications only attain standard status later in the life-cycle, following approval of the FIPA Architectural Board and the membership. TC indicates a Technical Committee.

FIPA believes in trying out the specifications in the field (experimental status) before they attain standard status. To the best of authors' knowledge, no specification has yet reached standard status, the most mature ones are only currently only experimental. This gives FIPA the option to investigate several potentially overlapping designs and then to adopt the most successful practical one.

FIPA specifications are developed and maintained on a continual basis. They are primarily discussed, reported and assigned a status (see below) at quarterly FIPA forum meetings. See [1] for details of the previous and future meetings. Non-member companies and organisations can attend FIPA meetings when contributing to FIPA's business.

## 5 Developing, Deploying and Implementing FIPA Specifications

FIPA specifications and the open source implementations have developed considerably to bring agent technology to world-wide deployment, enabling and promoting such projects as Agentcities. However, the FIPA specifications are not intended to be a complete blueprint or specification for building multi-agent system. For example, FIPA standards do not prescribe how to describe existential aspects of how agents in a discrete world, nor do they define error handling although some aspects of error reporting are covered. Some of the practical issues in using the FIPA standards are reported in [20]. Note that currently, there is no formal or clear mechanism to determine the compliance of a FIPA architecture implementation other than testing the interoperability of heterogeneous FIPA platforms in the field.

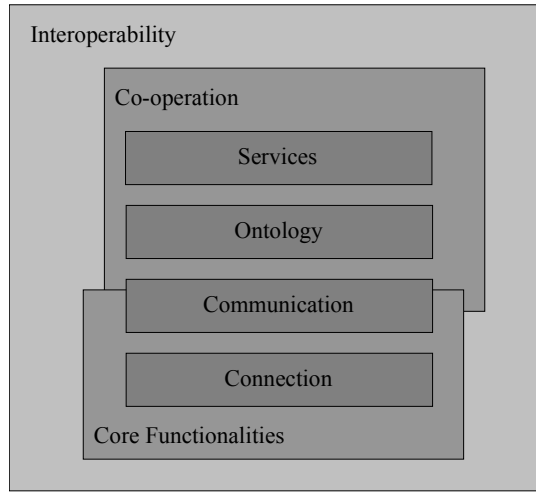


**Fig. 6.** Identification of interoperability layers [17]

Figure 6 summarises the interoperability layers that can be identified. From an interoperability perspective there are two key separations: core functionalities and co-

operation. The core-functionalities for the design and development of agent and distributed systems are well understood and are dealt with by a number of Open Source platforms. A particular event that has harden these interoperability checks was at the London FIPA meeting in 2001 [1] , were a parallel session of testing between three platforms occurred: JADE [21], FIPA-OS [22] and ZEUS [23]. The result was two-fold: a) Illustrated the advancement of the FIPA specs as interoperability was more efficient and was able to test more of the specifications than the previous event in the 1999 meeting and b) FIPA specifications were considered fairly thorough, in that, only minor changes were necessary to enable the "core functions" to be tested.

To move towards the more application level of smart services requires the co-operative aspects to be addressed and this is known to be complex. Some progress has been made in this area, examples of this are seen in the specification of the DF, service ontology structure and protocols for service registrations in the FIPA specification [12] and hence the systems can perform conformance testing for interoperability at this level.

### 5.1 Concrete Architectures

There is inherent high complexity, cost and risk in developing a good practical standard and in developing systems that adhere to the standard. These act as a barrier to users and developers who wish to assess this new technology. Clearly, reference implementations are useful here and there seems to be (at least) two main approaches to develop these. Essentially, reference implementations can be developed under an open license or a closed license. We focus on the former approach.

An open source licence implementation can reduce the barrier for the adoption of FIPA standards, enhancing the ability of agent application developers to construct applications using FIPA technology. Minimizing cost is particularly important to encourage take up in education establishments and small medium enterprises. It can also improve software quality through third party development - more eyes, less bugs [24]. Users can directly reduce the time-scale for bug correction by providing their own fixes- this effect can be dramatic within a large user base. Open source is a powerful mechanism for engaging users while the market for FIPA agent technology is developing – it can be regarded as a form of Rapid Application Development in which new user requirements and actual extensions can be incremented during the development process. The April Agent Platform [25], FIPA-OS, JADE, and ZEUS are FIPA based agent systems released under an open source license and provides rich and flexible support for agent communication. These open source projects can be regarded as the first concrete realisations of the FIPA specifications.

Two additional initiatives to use the specifications are also worth mentioning Firstly, the JAS [26] project is developing a concrete realisation of the FIPA Abstract Architecture – this is work in progress at this time. Secondly, there are number of collaborative projects that are using, applying and evaluating the FIPA specifications – a selection of these are discussed in the next section.

### 5.2 Collaborative FIPA-Based Projects

There are a number of projects that have used the FIPA specifications and open source platforms for development, we summarise only a few of these activities here (see the FIPA web-site for more examples):

- The goal of the FACTS [27] project was to validate the work of FIPA and other standards groups by constructing a number of demonstrator systems based on FIPA's proposed standards. The focus of the project was on the interaction between different implementations of agents and agent platforms. The project was structured around two development cycles: during phase 1, agent interoperability was tested primarily within each of three application areas (audio-visual broadcasting and entertainment, service reservation, electronic commerce); during phase 2, agent interoperability was tested between the different application areas.
- LEAP [28] is the precursor of the second generation of FIPA agent platforms. Project LEAP (Lightweight Extensible Agent Platform) addresses the need for open infrastructures and services that support dynamic, mobile enterprises. It will develop innovative agent-based services supporting three requirements of a mobile enterprise workforce: Knowledge management (anticipating individual knowledge requirements), decentralised work co-ordination (empowering individuals, co-ordinating and trading jobs), travel management (planning and co-ordinating individual travel needs). It represents a major technical challenge – and has become the first integrated agent development environment capable of generating agent applications and executing them on a wide variety of run-time environments (implemented on devices such as computers, PDA and mobile 'phones) and communication mechanisms (TCP/IP, WAP etc.).
- The CRUMPET [29], Creation of User-friendly Mobile services PErsonalised for Tourism, Project takes advantage of integrating four key emerging technology domains and applying to the tourism domain: personalised services, multi-agent technology, location-aware services and transparent mobile data communication. CRUMPET is also a second generation FIPA agent platform based on FIPA-OS and adapted to small foot-print devices such as PDAs and to communication over wireless links.
- The Agentcities project [15] is to help realise the research and commercial potential of agent applications and accelerate the deployment of next generation Internet services by: (a) Deploying an infrastructure: building a standardised, publicly accessible, continually available, pan-European network of agent platforms providing the necessary connectivity and infrastructure to host agent-based services, (b) Deploying services: populating this network with a rich assortment of commercial grade agent-based services to form building blocks for advanced agent applications, and (c) Fostering research collaboration: promoting the network as a Europe wide focal point for agent research and development, thereby enabling cross-fertilisation between existing and future projects.

The sample projects outlined above each bring a new demand on the interoperability requirements of the FIPA specifications and delve further into the semantics of the co-operation interoperability aspects. It is very clear that in order to

get a project such as Agentcities to work means that the semantics of ontologies, service interaction, commitments, trust etc. will have to be defined both as a specification and a design.

## 6 Conclusion and Further Work

The primary interoperability trials that have been performed between FIPA platforms within different projects and FIPA itself has pointed out differences between the implementations at all layers. A drawback of FIPA standards is that there is currently no means to certify a FIPA compliant platform. The main reasons for this certainly comes from the difficulty to validate platform behaviour through an interface that only allows messages to be exchanged. Another primary difficulty to perform validation tests is the need of a reference platform, which has to be defined by the standard bodies.

However, it is worth noting that FIPA has made major progress in advancing interoperability and addressing the issue of conformance testing. This concern of dealing with interoperability indirectly addresses some aspects of classical conformance testing, that is, at least the agent systems engaged in the interoperability tests conform in the same manner with FIPA specifications. The problem is that classical conformance testing in systems that exhibits the properties of autonomy and responsive behaviours, brings computational solutions to handling dynamic behaviour that cannot easily follow the classic model of conformance testing. In the dynamic and autonomous requirements of agent systems, we cannot rely upon "looking inside" of the agent system and analysing its computational behaviour. A company, business or end user will not want their communication strategy to be revealed. If we wish to trade the verifiability of an agent for autonomous behaviour then we need to consider other ways to test verification and conformance.

The key to moving in this direction of new modes of verification and conformance to allow the agent systems to be realistically applied to e-commerce's world is through considering social and organisational models. FIPA starts to address these organisational concerns through the domains and policies of models within the abstract architecture refinement. However, behaviour interpretation of these autonomous entities within a highly distributed environment requires a "social behaviour model". Part of the realisation of this is through the communication language and the semantics of commitments that FIPA starts to address within the technical committee for agreements. This process will expand the agent communication language that is currently specified within FIPA. However, this is not the whole story for realisation of right and wrong behaviours within the dynamic autonomous environment that is required for advancing the e-commerce's world. There are two other key features: models of trust and knowledge sharing.

Often in systems, the focus on concepts of trust is at the hard security level, and concerns theories of encryption and identification. However, there is a major difference between knowing that the data has been delivered without any interference and knowing that the entity that supplies that current data is reliable. The second issue of knowledge sharing sits under the hat of ontologies. Currently FIPA does not exploit some of the major advantages from having an ontology explicitly modelled

and available. It still uses a reference model. Also, knowledge sharing and knowledge interoperability within agent systems has never yet been attained beyond any simple level. The next phase of FIPA will start to consider what features can be specified to advance the semantics of "social behaviour" and its convergence with domains, policies and commitments.

## Acknowledgements

We wish to thank all the FIPA membership and collaborators, past and present, as the development and maintenance of the FIPA agent specifications discussed in this article is a joint effort. The opinions expressed in this article are those of the authors and do not necessarily reflect those of the FIPA membership at large. Our special thanks go to Stefan Channing at Queen Mary whose valuable comments helped to improve this paper.

## References

1. FIPA, The Foundation for Intelligent Physical Agents, home web-page, http://www.fipa.org
2. Franklin, S., Graesser A.: "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents". In: Proceedings of Agent Theories, Architectures, and Languages, ECAI workshop, Budapest, Hungary, August, (1996) 193 - 206
3. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. In: Journal of Autonomous Agents and Multi-Agent Systems. Vol. 3 No. 3 (2000) 285-312
4. Nwana H.S.: Software agents: an overview. Knowledge Engineering Review, Vol. 11, No.3, (1995) 205-244
5. Wooldridge, M., Jennings, N.R.: Intelligent agents: theory and practice. Knowledge Engineering Review, Vol. 10, No.2, (1995) 115-152
6. FIPA Abstract Architecture Specification. FIPA00001. FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents (2000)
7. FIPA Agent Message Transport Service Specification. FIPA00067. Foundation for Intelligent Physical Agents (2000)
8. FIPA Request Interaction Protocol Specification. FIPA00026. Foundation for Intelligent Physical Agents (2000)
9. Bauer, B., Mueller, J P. and Odell J: Agent UML: A formalism for specifying multiagent software systems. In: Ciancarini, P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering — Proceedings of the First International Workshop (AOSE-2000). Springer-Verlag, Berlin, Germany (2000)
10. Pitt, J., Kamara, L., Artikis A: Interaction Patterns and Observable Commitments in a Multi-Agent Trading Scenario. 5th Int. Conf on Autonomous Agents, Montreal, Canada (2001)
12. FIPA Agent Management Specification. FIPA00023. Foundation for Intelligent Physical Agents (2000)
13. Wooldridge, M.: Semantic Issues in the Verification of Agent Communication Languages. In: Journal of Autonomous Agents and Multi-Agent Systems. Vol. 3, No. 1 (2000) 9-31
14. Singh, M.: Agent Communication Languages: rethinking the principles. IEEE Computer. Vol. 13, No. 12, (1998) 40-47
15. Agentcities. Testbed for a Worldwide Agent Network: EU Research and Development Project, IST-2000-28385, http://www.agentcities.org/ (2001)

16. Poslad, S., Buckle, P., Hadingham, R.G.: Open Source, Standards and Scaleable Agencies. Autonomous Agents 2000 Workshop on Infrastructure for Scalable Multi-agent Systems, Barcelona, June, (2000)
17. Charlton, P., Bonnefoy, D., Lhuilier, D.N.: Dealing with Interoperability for Agent-Based Services, Autonomous Agents, Montreal, Canada, (2001)
18. Searle, J. R.: Speech Acts. Cambridge University Press Cambridge, UK (1969)
19. FIPA ACL Message Structure Specification. FIPA00061. Foundation for Intelligent Physical Agents (2000)
20. Charlton, P., Cattoni, R .: Evaluating the Deployment of FIPA Standards when developing Application Services, IJPRAI, Vol. 15, No. 3, (2001)
21. Bellifemine, F., Rimassa, G., Poggi, A.: JADE - A FIPA-compliant Agent Framework. In: Proceedings of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, London (1999)
22. Poslad, S. J., Buckle P., Hadingham R.: The FIPA-OS agent platform: Open Source for Open Standards. Proceedings of PAAM 2000, Manchester, UK, (2000) 355-368
23. Nwana, H., Ndumu, D., Lee, L., Collis, J.: ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems. In Applied Artifical Intelligence Journal, Vol. 13, No. 1, (1999) 129-186
24. Cathedral and Bazaar. HYPERLINKhttp://www.tuxedo.org/~esr/writings/cathedral-bazaar/
25. April Agent Platform. HYPERLINKhttp://sourceforge.net/projects/networkagent/
26. Java Agent Services, JSR-000087, HYPERLINKhttp://www.java-agent.org.
27. FACTS: FIPA Agent Communication Technologies and Services, EU Project Number ACTS-1997- AC317. http://www.labs.bt.com/projects/facts/
28. LEAP: "Lightweight Extensible Agent Platform", EU Project Number IST-1999-10211, http://www.cordis.lu/ist/projects/99-10211.htm.
29. Poslad, S., Laamanen, H., Malaka, R., Nick, A., Buckle, P., Zipf, A.: CRUMPET: Creation of User-friendly Mobile services PErsonalised for Tourism. 3G2001 IEE conference, London, (2001)