Review of FIPA Specifications

Stefan Poslad (email: FIPA-ROFS-Chair@ieee.org). Other authors and affiliations to be added

Version	Date	Notes
0.1		Outline of document structure
0.2		Content for section 2 added
0.3	2006-02	Content for section 2 expanded
0.4	2006-02	Added note about FIPA representations section 2.6 expanded section 3 and 4 a bit more
0.6	2006-07	Section 3.7 started and other sections in 2 and 3 have minor modifications
0.7	2006-9	Restructured. Section 3 added about deployed FIPA tools and applications. FIPA section 4 on FIPA models and constraints nearly completed. Section 5 on specifications that did not make it to standard started and is about 50% complete
	2006-12	Complete Review.

Note as this is still a draft, this document has not been proof-read.

Abstract

The purpose of this report is to present a critical analysis of the FIPA Multi-Agent System (MAS) specifications and to highlight areas for further work. This presents an in-depth study that considers the FIPA specifications as a whole. It provides a holistic and detailed discussion of the complete range of specifications. It lo presents the design insight, of the design choices of the FIPA specifications that were considered, and why the ultimate design trade-offs were made.

Table of Contents

1 Introduction

1.1 Motivation

FIPA was established in 1996 as an international non-profit association of companies that agreed to share efforts to produce standard specifications of generic agent technologies that were: produced in a timely fashion, internationally agreed and usable across a large number of applications so that a high level of interoperability across applications is achieved. Since then, FIPA has counted more than 60 members from more than 20 different countries-worldwide and generated a set of specifications that went through 3 cycles of review: FIPA97, FIPA98, FIPA2000. Several distinct agent platforms, applications, and collaborative projects have been, and are continuing to be, based upon the FIPA specifications; the core set of the specifications have been used for a number of years and they are robust and effective enough to be promoted to Standard. The X2S TC was created at the 24th FIPA meeting in Lausanne, Feb. 2002, to drive standards to standard status, harmonize, ensure coherency, correctness. It went through 3 iterations of accepting comments from the membership and improving the specifications. At the end of 2002, the FAB believed that these specifications were now stable, mature, well understood and ready for commercial implementation and deployment. In 2005, FIPA became the 12th IEEE standards activity. More detailed background for FIPA can be found in [Poslad, 2005].

The purpose of this report is to present a critical analysis of the FIPA Multi-Agent System (MAS) specifications and to highlight areas for further work. There is a wide body of existing work that has discussed specific FIPA specifications; that has identified limitations of FIPA specifications and has made suggestions for modifications or alternative models. However, there has been a lack of studies in depth that has considered the FIPA specifications as a whole, that has classified and analysed the individual FIPA research papers together. There is no reported work that has taken a holistic and detailed discussion of the complete range of specifications. There are no reports that have presented the design insight, of the design choices of the FIPA specifications that were considered, and why the ultimate design trade-offs were made. There is no analysis of how the FIPA models relate to those from other overlapping and competing standards bodies. There is no assessment of the current scope and status of the specifications with a view to setting out a road-map to maintain and enhance the FIPA specifications. There is no complete and general discussion of the main assumptions and features (vs. problems) in the FIPA specifications. If users of the FIPA specifications better understood the features and their limitations, they could more effectively deploy FIPA agent systems. Hence part of the motivation for this review is to address these concerns.

The other motivation for this review is to propose that the FIPA MAS models are as relevant and as valuable for modelling heterogeneous distributed computing services today as they have been in the past. In fact, it can be argued that the potential of the FIPA MAS model is greater today than it is been in the past because only now do we have the basic ubiquitous computing and computing infrastructure for multiple-agents to live in and real and powerful drivers to deploy them. The drivers for multi-agent systems are that the world is becoming more interconnected, using more functionally complex networked digital devices that are more interoperable, that are far more heterogeneous, at least at the application layer.. This is leading to the challenges of the interoperability of heterogeneous elements; drive from networked to service-oriented models for distributed computing; the widespread use of more embedded AI models in word-processors in (OCR) scanners, synthetic speech input and output devices, etc. Challenges of semantic interoperability, emergent behaviours, self organising systems … bla bla bla TODO complete this. Give an illustrative example here … e.g., a 21st century bus timetable for mobile users?

1.2 Agent properties and current scope of FIPA specifications

The essence of what constitutes an agent varies. Rather than define what is an agent and what constitutes a MAS and specifically a FIPA MAS, the range of a number of different agent and MAS properties or dimensions will be explored and FIPA MAS will be positioned according to these dimensions. [Sing et al. 2005] has classified multi-agents along multiple dimensions see, Figure 1. This provides a useful reference in order to specify the scope of the current FIPA multi-agent models. Future work may include further agent properties. Generally, two main types of agent properties can be differentiated: single or internal agent vs. multiple or external agent properties. Generally FIPA focuses on specifying external or MAS behaviour rather than on individual or internal agent behaviour, however, these are connected,

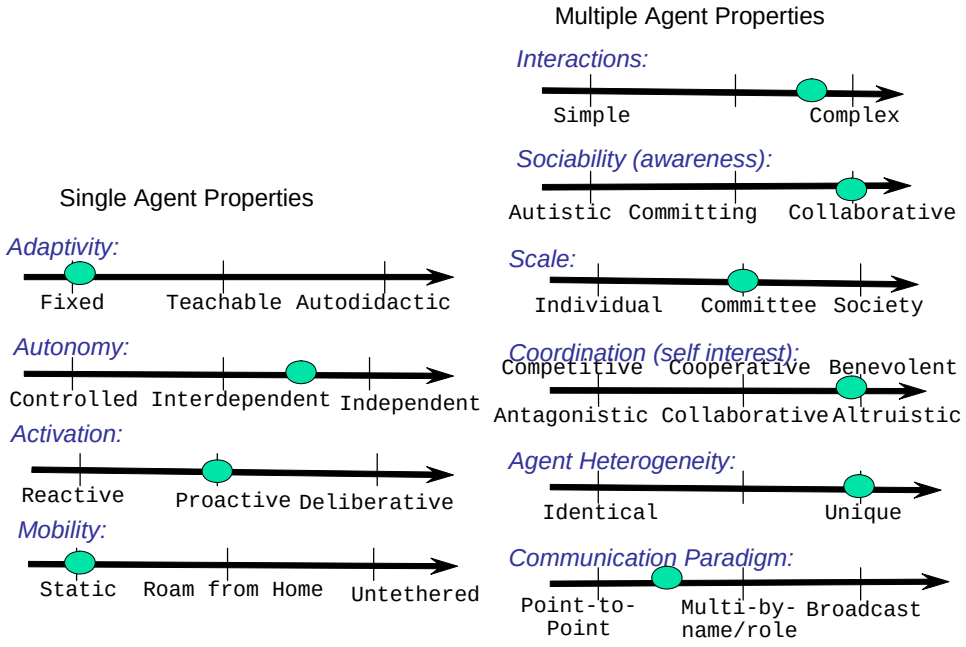


Figure 1: Classification of FIPA MASs according to the dimensions specified, adapted from [Sing et al/. 2005]. The shaded circle indicates the point on the scale that FIPA is deemed by the authors of this paper to operate on

1.2.1 Individual and internal agent properties

Activation (reactive and proactive vs. deliberative) of agent: this range from reactive agents that are passive and are activated externally when input events match conditions that then trigger actions, to proactive agents activated by internal (e.g., timer) events that match conditions and then trigger outputs, to deliberative agents that have their own internal activation loop and internal model of their operation and environment and can reason about these. FIPA specifications support reactive and proactive activation in order to support its CA communication and IP or Interaction Protocol. FIPA does not specify the process of deliberation as this is regarded as an internal agent property.

Adaptivity of an agent is the ability of agents to learn. This ranges from fixed (no learning), through to teachable, for example supervised learning through to autodidactic, e.g., unsupervised learning. Learning is regarded as an internal agent property so FIPA specifications support static adaptivity.

Autonomy is an agent property that ranges from independent, through to interdependent to controlled. FIPA agents most relate to interdependent as agents typically interact with organisation that fixes their interdependencies. For example, the FIPA agent management specification [FIPA0023] requires agents to interact to register with an agent management agent and promotes agents to interact with a directory agent to register their services. If agents interact using a FIPA Interaction Protocol, agents become interdependent.

Uniqueness: agents can range from being homogeneous to heterogeneous. In general, there are often several types of agent, performing different roles within a typical application domain. Hence FIPA MAS consist of heterogeneous agents.

1.2.2 Multi-agent or external agent properties

Coordination: this ranges from being competitive, cooperative (in between competitive vs. benevolent) to being benevolent where agents put their joint commitment to complete a joint interaction above their own plans. The underlying formal semantic model of the FIPA communicative action model, see section 2.2, indicate that FIPA agents are benevolent. However, when the FIPA interaction protocol models are considered instead, FIPA agents can cancel interactions, refuse or fail interactions supporting a less benevolent approach.

Interaction (complexity): these rage from simple (such as request-reply) direct, pull-type two-party and indirect (via a third party) interactions to more complex multi-party interactions. FIPA supports a range of complex interactions, see Table 2.

Interaction (cardinality): agents typically interaction between pairs of single agents, e.g., a user and provider of agents. However some FIPA interaction involve mediators and are between multiple parties, e.g., the Contract Net task-sharing protocol, see Table 2.

Sociability (awareness): this ranged from no interest in others (autistic) to being aware, to being a highly collaborative. FIPA MAS supports a collaborative type of interaction based upon the formal semantic model that define the CA messaging.

Scale: the number of agents range from individuals, through committees (~ <100) of agents to societies (~ 1K – M) agents. FIPA MAS systems o date tend to be committees in number. Some more ambitious projects such as the Agentcities.RTD project [give REF]have included of the order of 150 heterogeneous MAS, each with normally less than 1 agents, however these platforms were clustered according to the problem they solved and clusters of platforms tended to be small 2-5. The scale of FIPA systems is limited by mechanisms that FIPA specifies to federate MASs as specified in the FIPA agent management specification [FIPA0023].

1.2.3 (Currently) Out of scope agent properties

Other dimensions not included in FIPA specifications include agent properties of adaptivity, type of deliberation or reasoning, mobility, other types of MAS properties such as commitments, types of software architecture for agents, e.g., subsumption, BDI, social and market models etc, other and any types of human-agent interaction.

For properties, not supported in the standard FIPA specifications, these can be added in extensions to FIPA MAS frameworks. TODO- look for examples.

1.3 Contents Overview

Section 2 focuses on the different model viewpoints and main uses of FIPA Specifications and the FIPA MAS Model. These include: the network protocol stack view with multiple sub-layers at the application layer; the services architecture view, the distributed BDI model, the communication act language view etc.

Section 3 describes deployed systems, tools and applications. It focuses on design maturity, deployment issues, survey of deployments. It talks about the criteria for specifications maturity and how well FIPA fits these. For example tools and methods are a sign of software maturity.

Section 4 focuses on the constraints, features (vs. problems) in the FIPA specifications.

Section 5 describes some activities and specifications that were initiated that did not result in standard specifications being produced. The features and limitations of the models proposed are considered.

Section 6 gives the relationship between FIPA and other standards bodies. Standards are created to enable a critical bunch of stake-holders to agree on a set of specifications, for example to meet particular, new information or communication requirements for developing technological solutions. FIPA like most standards specifications is not able to support all conceivable vertical markets.

Section 7 discusses how to maintaining and enhancing the FIPA specifications. A key question here is whether or not an evolutionary path should be defined for some new activities versus a revolutionary path for other activities; whether or not we draw a line over the existing specifications vs. doing some maintenance to increase the utility and take up of the specifications; need to consider about the audience of roadmap, i.e., is it aimed at developers rather than business service managers. It makes recommendations and gives conclusions.

2 Viewpoints of the FIPA Specifications

The set of FIPA specifications can be viewed and utilised for a number of different purposes; these are described in this section, 0. For each view or purpose both the concepts of the view and the representation of the view are described. The FIPA model to date has focuses more on explicitly specifying how agents communicate and connect and less on specifying components such as agents, humans, data and services that transform, generate, process (including reason about) and store messages that are communicated, Figure 2.

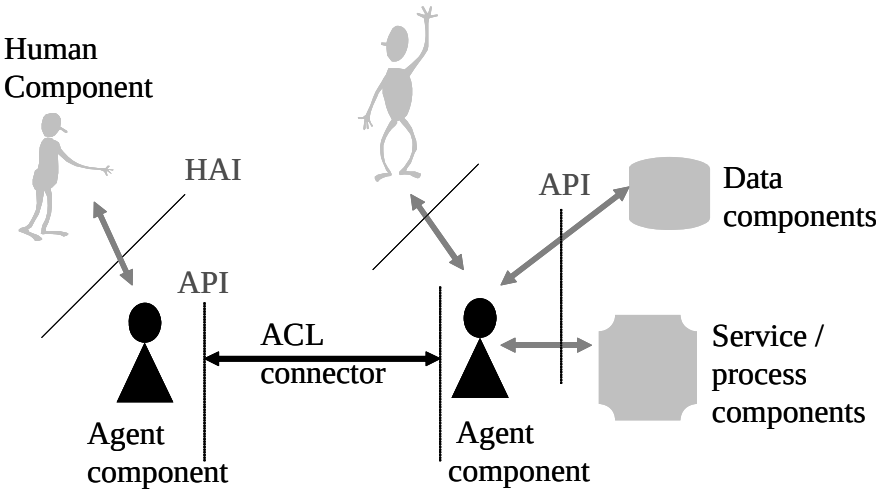


Figure 2. Abstract architectural model for a multi-agent system specified in terms of connectors and components.

The focus on this section is on specifications that matured into standards. These were mainly focussed on elements of the agent communication and to lesser extent on services. Some specifications of interest such as for Human-Agent interaction are discussed further in Section **Error! Reference source not found.**.

2.1 Layered communication protocol view

At a high-level of abstraction, a distributed computing model is composed of two parts: connectors (network) and (computation or service) components. Components are consumers, producers and mediators of communication messages. The early standards bodies such as ISO and IETF were network-oriented, resulting in layered protocol stacks such as the OSIRM (Open Service Interconnection Reference Model) and the TCP/IP Model and associated specifications. These protocols can be accessed via interfaces to services (software) that implement these protocols.

In the 1990s, service-oriented models, for example from organisations such as OMG, DCE, W3C, GGF and FIPA were developed and specified to supplement network-oriented models. A service oriented model can also be viewed as a communication protocol stack but uses multi sub-layer application protocols instead of a single layer application protocol, see Figure 3. This does not represent a strict layered protocol where a layer above can only be accessed by a layer below. The purpose of each of the sub-application layer protocols is as follows.

Sub-layer 1 (Transport): In the FIPA-ACL layered protocol model, the lowest application sub-layer protocol is the transport protocol. An asynchronous way of using the synchronous oriented HTTP is

specified in [FIPA0084]. Note that other standards bodies such Open Mobility Alliance (OMA) have also specified asynchronous ways of using HTTP.

Sub-layer 2(Encoding): Rather than send byte encoded messages, higher level data structures are encoded and transported, e.g., XML specified in [FIPA0071]. In part, because of the verbosity of XML, additional message encodings can be used instead of XML such as a string encoding [FIPA0070] and bit-efficient encoding [FIPA0069]. The latter is of particular use over lower bandwidth network links such as wireless links.

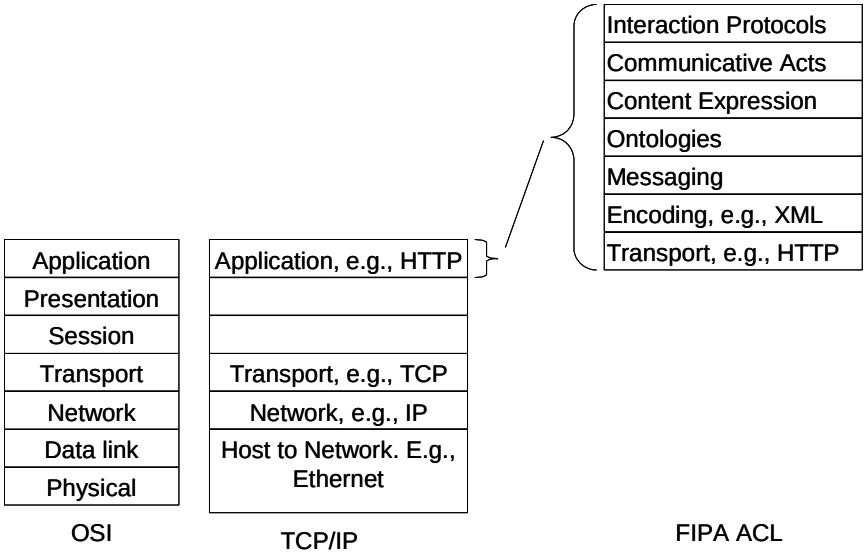


Figure 3. The FIPA-ACL protocol 'stack' and its relation to the TCPIP and OSI protocol stack

Sub-layer 3(Messaging): Message exchange requires the specification of data parameters in addition to the payload or content that is exchanged, e.g., the sender and receiver names bound to network addresses, the message type (FIPA Communicative Act type), time-outs for replies etc. A simplified example FIPA-ACL message structure is given in Figure 4. Unlike W3C-SOAP, the messaging structure is specified independently of the encoding, see [FIPA0061].

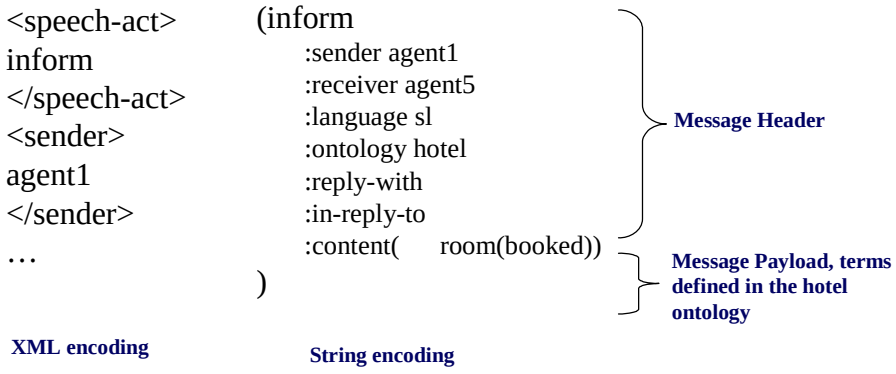


Figure 4: Simplified (not all fields are shown) message structure of FIPA-ACL messages encoded in String or XML format.

Sub-layer 4 (Ontology): the terms in the payload or content of an agent message are explicitly referenced with respect to an application-specific conceptual model, e.g., in Figure 4, the hotel ontology is used. Web resources may be used to identify Ontologies. FIPA does itself specify any particular representation for the Ontology, e.g., W3C-OWL, or any particular level of expressivity for Ontologies. Ontologies ranging from so called weak Ontologies such as taxonomies to strong Ontologies that also support logical inferencing can be used.

Sub-layer 5 (Content expression): FIPA defines general logical formulas and predicates that compute truth values and algebraic operations for combining and select concepts but separates theses from the use of defined application concepts (the Ontology). The former is specified using a content expression language, whereas the latter is specified by the (application) Ontology language. Examples of logic formulae include: *not, or, implies, equiv* etc. Examples of algebraic operators include *any* and *all*. This is discussed further in Section 2.2.1. FIPA has standardised one type of content expression, FIPA-SL, in [FIPA00008]. Note other types of content expression have also been specified such as W3C-RDF and a constraint language but these did not mature into standard specifications.

Sub-layer 6 (Communicative Act or CA): this is the core message protocol of FIPA, specified in [FIPA0037]. Some messaging protocols such as HTTP are minimalistic defining a small number of communication primitives or message types based upon a client-server type request (get) response (post) protocol However, the FIPA CA protocol is much richer than this. This defines communication in terms of a functions or actions, called the Communicative Act or CA, performed by the act of communicating. Additional parameters are needed to define CAs, for example a request action type CA requires a sub-action such as register in order to communicate a request to register a service description. This is discussed in more detail in Section 2.2.

Sub-layer 7 (Interaction Protocol or IP): typically in a service-oriented distributed computing model, messages are not exchanged in isolation but are part of an interaction sequence for work-flows, information exchange and task delegation. FIPA has standardised several interaction protocols, [FIPA0023], [FIPA0026], [FIPA0037], [FIPA0028], [FIPA0029], [FIPA0030], [FIPA0001], [FIPA0035] and [FIPA0036], see **Error! Reference source not found.**. This is discussed further in section 2.3.

2.2 CA or Agent Communication as Actions Model

The essence of the FIPA MAS model is that it specifies agent communication in terms of an Agent Communication Language (FIPA-ACL). The FIPA-ACL defines communication in terms of a function or action, called the Communicative Act or CA, performed by the act of communicating. There are many differing ranging view of the types of communication that characterise agent communication. For example, agent use of mentalistic CA to exchange attitudes in terms of beleifs and intentions, the use of Meta-conceptual CA for knowledge exchange, the use of directives and assertives for agents acting as more specialised distributed computing entities, the use of agents as human proxies that use expressive CA, the use of agents to mediate etc. There are varying classifications for these functions that overlap but the one here seeks to combine models postulated by Jakobson (1963), Huhns (1974) and Singh (1998) to support a more general and flexible view of agent communication that can include all the various specialised types of agent communication. The set of types of action or functions includes:

1. *Connative* or *Directive* functions where a sender gives directives to the receiver to perform actions such as processing data. This type of function is most often used to initiate communication in place of Phatic functions.

1. *Directives*: that request, query or constrain requests or queries.

   a. *Interrogatives* / *Queries*: query another agent for information. This is equivalent to a get operation.
   b. *Exercitives* / *Delegation*: ask another agent to carry out an action
   c. *Permissives*: give permission to another to act on an object
   d. *Prohibitives*: withhold permission to another to act on an object
2. *Mediate*: acting as an intermediary between two participants to pass on information and tasks.
3. *Referential* or *Assertive* functions that a sender uses to share assertion type information about the world with receivers. This is equivalent to a set operation.
4. *Phatic* function that serve to establish (e.g., hello or open channel), prolong, interrupt communication (pause or close channel) or to check if communication is working; these also are akin to speech utterances that support politeness in human speech
5. Temporal functions
6. *Commissive*: function that promise the commitment to some future action
7. *Meta-linguistic* functions that allows messages to be related to other messages or concepts.
   a. *Para-linguistic*: a message is related to other messages (already sent or about to be sent)
   b. *Meta-conceptual or semantic*: a message is related to other shared concepts such as those in one or more domain specific ontologies.
   c. *Contextual*: a message is associated with saying something about the time, place, or persons in the interaction. Many linguistic forms referring to these things cannot be interpreted without reference to the speech act itself, for their meanings are not fixed but relative (e.g, here, there, now, then)
8. *Expressive*: express emotions and attitudes toward receiver that are generally under voluntary control of sender
   a. *Mentalistic*: functions that express the attitudes, in terms of intentions and beliefs, of the message sender to receivers.
   b. *Poetic / Emotional*: function is expressed as restrictions on message form of many different sorts such as different degrees and varieties of aesthetic pleasure are derivable from various ways of formulating a message with any given referential content.
   c. *Rhetorical*: acts that are issued to create an effect in the receiver without expecting or needing an answer
9. *Declarative*: that causes events in themselves, that have a wider significance in society
   a. *Preservative*: control antecedently existing activities, e.g. traffic regulation,
   b. *Constitutive*: create or constitute the activity itself, e.g. the rules of the game.

The set of FIPA standardised CAs from [FIPA0037] are classified with respect to the type of action in Table 2. A message can perform several functions at the same time. For example the FIPA CA *Agree* is described as the action of agreeing to perform some action possibly in the future. This is phatic in terms of agreeing to proceed and is para-linguistic in terms of referring to another FIPA CA. The classification highlights the principal function of the message from secondary functions - other

functions carried by the message. Hence, Agree is principally classified as a Phatic function. All FIPA CAs inherently support the expressive function as a secondary function as they are defined in a modal logic form that expresses attitudes, intentions and beliefs, see section 2.2.1. All FIPA CAs support the meta-conceptual function: they can refer to concepts from an explicit conceptualisation defined in an explicit conceptualisation such as an Ontology, see Section 2.2.2 . There are several other classifications of CAs, e.g., Searle [ref], these differ in terms of the functions defined, e.g., Searle defines a Promissive function that defines actions to be performed in the future: this could be loosely equated to the paralinguistic function. Searle defined: declarative function that performs an act merely by issuing the utterance "I sentence you to two years' imprisonment." Permissives and Prohibitives type functions are not supported or standardised as FIPA-CAs.

Communicative Act (CA)	description
accept-proposal	act of accepting a previously submitted proposal to perform an act
agree	act of agreeing to perform some act, possibly in the future
cancel	act of cancelling some previously requested act which has temporal extent
cfp	act of calling for proposals to perform a given act
confirm	S informs R that a given proposition is true, where R is known to be uncertain about the proposition
disconfirm	S informs R that a proposition is false, where R is known to believe, or believe it likely that, the proposition is true
failure	act of telling another agent that an act was attempted but the attempt failed.
inform	S informs R that a given proposition is true
inform-if	act for the agent of the act to inform the recipient whether or not a proposition is true
inform-ref	act for the S to inform R the object which corresponds to a descriptor
not-understood	S informs R that it perceived that B performed some act, but S did not understand what B did
propagate	S intends R to treat embedded message as sent directly to it, and wants R to identify the agents denoted by the given descriptor and send the received propagate message to them
propose	act of submitting a proposal to perform a certain act, given certain preconditions
proxy	sender wants the receiver to select target agents denoted by a given description and to send an embedded message to them
query-if	act of S asking R whether or not a given proposition is true
query-ref	act of S asking R for an object referred to by a referential expression
refuse	act of refusing to perform a given act and explaining the reason for the refusal
reject-proposal	act of rejecting a proposal to perform some act during a negotiation
request	S requests R to perform some act.

request-when	S wants R to perform some act when some given proposition becomes true
request-whenever	S wants R to perform some act as soon as some proposition is true and thereafter each time the proposition becomes true again
subscribe	act of requesting a persistent intention to notify S of the value of a reference, and to notify again whenever the object identified by the reference changes

Table 1. FIPA ACL message types or Communicative Acts (CAs)

Communicative Act (CA)	Base CA	Assertive	Query	Mediate	Delegation	Phatic
accept-proposal	inform				X	
agree	inform					X
cancel	disconfirm					X
cfp	query-ref				X	
confirm	confirm	X				
disconfirm	disconfirm	X				
failure	inform					X
inform	inform	X				
inform-if	inform	X				
inform-ref	inform	X				
not-understood	inform					X
propagate	inform			X		
propose	inform				X	
Proxy	inform			X		
query-if	request		X			
query-ref	request		X			
refuse	disconfirm; inform					X
reject-proposal	inform				X	
request	request				X	
request-when	inform				X	
request-whenever	inform				X	
subscribe	Request-		X			

	whenever					

Table 2. FIPA Communicative Acts (CAs) classified according to the type of function or action of the CA. All FIPA CAs have their semantics defined using beliefs and intentions. Some CAs are defined as composite derived from other base CAs.

Agents that communicate using an ACL have flexibility because they can communicate to support functions 1-5. In contrast mainstream distributed computing services such as client-server type Web services focus most on communication functions 1a, 1b and 2.

2.2.1 CA Beliefs and Intentions Model

In this view, the sender expresses the semantics of messages it shares with receivers in terms of mentalistic models of beliefs and intentions. Consider the basic case shown in Figure 5, taken from Spec XXXX Suppose that, in abstract terms, Agent i has amongst its *mental attitudes* the following*:* some goal or objective ***G,*** some *beliefs* ***B*** and some intention ***I***. Deciding to satisfy ***G***, the agent adopts a specific intention, ***I***. Note that neither of these statements entail a commitment on the design of Agent i: ***G*** and ***I*** could equivalently be encoded as explicit terms in the mental structures of a BDI agent, or implicitly in the call stack and programming assumptions of a simple Java or database agent.

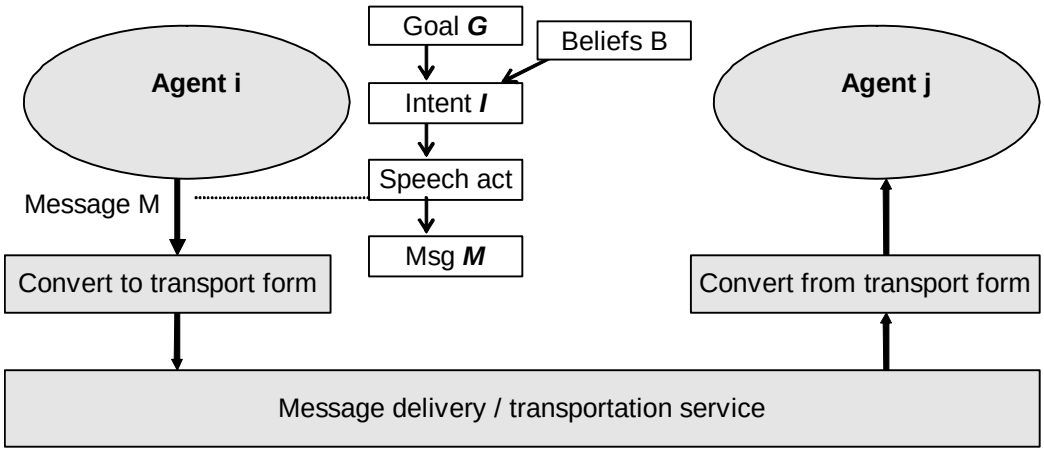


Figure 5: Sender expresses its mental attitude when communicating with a receiver

An agent *i* has *pg* as a persistent goal, if *i* has proposition *p* as a goal and is self-committed toward this goal until *i* comes to believe that the goal is achieved or to believe that it is unachievable. *Intention* is defined as a persistent goal imposing the agent to act. Formulas as $PG_i p$ and $I_i P$ are intended to mean that "*i* has *p* as a persistent goal" and "*i* has the intention to bring about *p*", respectively. The definition of *I* entails that *intention generates a planning process.* See Sadek (1992) for the details of a formal definition of intention.

Assuming that Agent i cannot carry out the intention by itself, the question then becomes which message or set of messages should be sent to another agent (j in *Figure 1*) to assist or cause Intent ***I*** to be satisfied? If Agent i is behaving in some reasonable sense rationally, it will not send out a message whose effect will not satisfy its intention and hence achieve its goal. Sender only acts in a way it hopes will satisfy his intention and meet his goal (assuming that sender thinks that the receiver will help). The sender can reason that the effect of asking receiver is that the receiver would tell the sender, hence making the request fulfils his intention. Now, having asked the question, can the sender actually

assume that, sooner or later, its intention will be answered? The sender *can* assume that receiver knows that it does not know, *and* that it knows that sender is asking the receiver to tell him. But, simply on the basis of having asked, the sender cannot assume that receiver will act to tell it the result of a query: the receiver is independent, and may, for example, be busy elsewhere.

In summary: an agent plans, explicitly or implicitly (through the construction of its software) to meet its goals ultimately by communicating with other agents, that is, sending messages to them and receiving messages from them. The agent will select acts based on the relevance of the act's expected outcome or *rational effect* to its goals. However, it cannot assume that the rational effect will necessarily result from sending the messages.

The Semantic Language (SL[1]) is used to define the semantics for the FIPA CAs as a logic of mental attitudes and actions, formalised in a first order modal language with identity[2], see [FIPA00037] for details of this logic. In order for a communicative act (CA) to be planned or intended by the sender, both (preconditions), the reasons for which the act is selected and the (post)conditions that should be satisfied when the act is completed, have to be specified. For a given act, the former is referred to as the *rational effect* or *RE*[3], and the latter as the *feasibility preconditions* or *FP*s, which are the qualifications of the act. For each CA, its semantics have been defined in [FIPA0037] in terms of the intentional state expressed by the sender agent and the associated RE and FP for selecting and intending the CA sent.

The following example, Figure 6 defines the sender expressed intention for a directive type CA request from a sender i to a receiver j:

<i, REQUEST (*j*, *a*)>

[1] SL is also used for the content language of the FIPA ACL messages (see [FIPA00008]).
[2] This logical framework is similar in many aspects to that of [Cohen90].
[3] Rational effect is also referred to as the *perlocutionary effect* in some of the work prior to this specification (see [Sadek90]).

FP: FP (*a*) [*i**j*] $\wedge$ B_i Agent (*j*, *a*) $\wedge$ B_i $\neg PG_j$ Done (*a*)

RE: Done (*a*)

Where:

a is a any action expression

FP (a) denotes the feasibility preconditions of *a*

FP (a) [i\j] denotes the part of the FPs of *a* which are mental attitudes of *i*.

p, p_1, ... are taken to be closed formulas denoting propositions,

i and *j* are schematic variables which denote agents

Done (*a*) means that *a* has taken place

Agent (*j*, *a*) is a proposition meaning *j* denotes the only agent that ever performs (in the past, present or future) the actions which appear in action expression *a*.

$\neg PG_j$ means that j does not (already) have the Goal or objective G of Proposition P in this case doing a already.

Figure 6. Example logic 1st order modal formula to define the request communicative act

Note that according to [FIPA0037], there are 4 primitive or atomic CAs inform, query, confirm and disconfirm are primitive. Other CAs are composite CAs, combined out of 2 or more primitive CAs using the or, and operators. For example, see Figure 7. A distinction made is that atomic CA can be directly carried out resulted in a done, whereas all other CAs can only be planned.

`inform-ref` act. `inform-ref` is a macro act

It is defined formally by:

<*i*, INFORM-REF (*j*, ι*x* δ(*x*))> ≡

<*i*, INFORM (*j*, ι*x* δ(*x*) = r_1)> | … | <*i*, INFORM (*j*, ι*x* δ(*x*) = r_n)>

where *n* may be infinite

i and *j* are schematic variables which denote agents

Figure 7. Example of a composite CA , e.g., inform-ref defined using one or more primitive CAs, e.g., inform.

This act may be requested (for example, *j* may request *i* to perform it) or *i* may plan to perform the act in order to achieve the (rational) effect of *j* knowing the referent of δ(*x*). However, when the act is actually performed, what is sent and what can be said to be `Done`, is an `inform` act.

Composite speech acts represent the ability to extend primitive CAs to synthesise new CAs from the primitive ones. Another form of extensibility is an inter-agent plan is a combination of such communicative acts, using either the conjunction, disjunction and sequence operators, used between two or more agents. FIPA interaction protocols are other models of pre-enumerated inter-agent plans, see section 2.4.

Communicative Act or function	SL semantics to define sender Intent
accept-proposal: S informs R it intends R to perform the proposal action when	<i,accept-proposal(j,<j,act>,ϕ))> ≡

the proposition precondition becomes true	<i,inform(j,IiDone(<j, act>, ϕ))> FP: Bi α ∧ ¬Bi (Bifj α ∨ Uifj α) RE: Bj α Where: α = Ii Done (<j, act>, ϕ)
agree: R informs S that it intends to perform the action but not until the precondition becomes true	<i,accept-proposal(j,<j,act>,ϕ))> ≡ <i,inform(j,IiDone(<j, act>, ϕ))> FP: Bi α ∧ ¬Bi (Bifj α ∨ Uifj α) RE: Bj α Where: α = Ii Done (<i, act>, ϕ)
cancel: S informs R that it no longer intends R to perform a previous requested action t	<i, cancel (j, a)> ≡ <i, disconfirm (j, Ii Done (a))> FP: ¬Ii Done(a) ∧ Bi (Bj Ii Done(a) ∨ Uj Ii Done(a)) RE: Bj ¬Ii Done(a)
cfp: act of calling for proposals to perform a given act	<i, cfp (j, <j, act>, Ref x ϕ(x))> ≡ <i,query-ref(j,Ref x(IiDone(<j, act>, ϕ(x)) ⇒ (Ij Done (<j, act>, ϕ(x))))> FP: ¬Brefi(Ref x α(x)) ∧ ¬Urefi(Ref x α(x)) ∧ ¬Bi Ij Done (<j, inform-ref (i, Ref x α(x))>) RE: Done (<j, inform (i, Ref x α(x) = r1)> | … | <j, inform (i, Ref x α(x) = rk)>) Where:α(x) = Ii Done(<j, act>, ϕ(x)) ⇒ IjDone(<j,act>, ϕ(x))
Confirm: S believes P is true, intends that R will believe P is true, believes that R does not already know if P is true or not.n	Etc.
Disconfirm: S believes P is false, intends that R will believe P is false, believes that R does not already know if P is true or not	
failure: S believes that R is capable of doing an act but didn't	
inform: S believes P is true, intends that R will believe P is true, believes that R does not already know if P is true or not.	

inform-if: S informs R whether or not it believes P is true.	
inform-ref: S informs R it believes that a match is found.	
not-understood: the sender A informs the receiver B that it perceived that B performed some act, but A did not understand what B did	
propagate: sender intends that the receiver treats the embedded message as sent directly to it, and wants the receiver to identify the agents denoted by the given descriptor and send the received propagate message to them	
propose: act of submitting a proposal to perform a certain act, given certain preconditions	
proxy: sender wants the receiver to select target agents denoted by a given description and to send an embedded message to them	
query-if: S ask R if it believes P is true because S does not know if P is true and beleives that R knows about the truth of P	
query-ref: S ask R for a matching IRE because S does not know a matching IRE and believes that R knows about a matching IRE.	
refuse: S informs R that act is infeasible, act is not done and no intention to do act	
reject-proposal: S informs R that because of P1, it does not intend for R to do act with precondition P2.	
request: S intends R to perform an act that it believes has the ability to do and intends to do the act	.
request-when: S intends R to perform an act when an event causes to R to believes a precondition is true	
request-whenever: S intends R to perform an act whenever an event	

causes to R to believes a precondition is true	
subscribe: act of requesting a persistent intention to notify the sender of the value of a reference, and to notify again whenever the object identified by the reference changes	

Table 3. Communicative Acts (CAs) expressed using a intentional model for sender S n a receiver R acts on a message from Sender S. See [FIPA00037] for an explanation of the logic used.

2.2.2 A Meta-linguistic CA Model

Specifying the semantics of a CA requires more than specifying the name and conceptualisation of the CA; the CA also needs to refer to other meta-concepts such as sender and receiver identities and addresses. There is a range of different meta-concept representations of different expressivity that could be used ranging from weak conceptual models without any logic formalism to stronger conceptual models with stronger logic formalism. Hence, there are two different ways to refine the CA model for it to be complete enough to be used as in practice: a weak meta-concept model, can be defined to model CA functions 1-4, see Section 2.2 . A stronger conceptual models with stronger logic formalism, e.g., [FIPA00037], is needed to define menatlistic type semantics.

	Meta concepts		
Communicative Action or function	.Proposition (P)	IRE	Sub-Action (A)
accept-proposal	Agreement Condition		A
agree	Agreement Condition		A
cancel			A is cancelled
cfp:	Contract Pre-conditions	Offer	A to be performed
confirm	R's P is true		
disconfirm	R's P is false		
failure	Reason for failure		A
inform	P is true		
inform-if	Tell R if P is true or not		
inform-ref		Result	
not-understood			A
propagate	Condition to propagate	Rs for message	A defined by S
propose	Pre-conditions		A
proxy	Pre-conditions	Rs for message	A defined by S
query-if	P sent to R to check if it		

	is true		
query-ref		Result	
refuse	Reason for refusing		A
reject-proposal	Original proposal, reason for rejection		A
request			A
request-when	Condition for A		A
request-whenever	Condition for A		A
subscribe		Result	

Table 4: FIPA ACL message types or Communicative Acts (CAs) specified in terms of the meta-concepts where S is the sender and R is the Receiver .

Several generic meta-concepts are used by all CAs such as the identities of the sender and receiver. In addition there are CA specific meta-concepts that need to be defined such as a delegation type directive requires a sub-action meta-concept, e.g., service Agent A requests broker agent B to do sub-action to publish of its service description Y in the broker's directory. In contrast, a query type CA needs to define a free variable, called an identifying reference expression or IRE, that matches a condition and algebraic type set operations to select a variable that matches. The meta concepts associated with each CA is defined in [FIPA SC000067] and is given in Table 4. In the general case, a weakly expressive conceptual framework that supports algebraic operations would suffice to define CAs for functions 1-4, given in Section 2.2. Note that sub-actions can also be further CAs: this gives the CA model *extensibility* and the ability to synthesise additional composite CAs..

Rather than just model the associated meta-concepts for each CA as a set of input and output parameters, FIPA has also modelled the associated meta-concepts more expressively and explicitly in a content expression language. FIPA separates logical formulas and predicates that compute truth values, algebraic operations for combining and select concepts, from the definition of the application concepts (the Ontology). The former is defined in a separate content express protocol language, the latter is defined in an application specific ontology language. Examples of logic formulae in an expressive content language include: *not, or, implies, equiv* etc. Examples of algebraic operators include *any* and *all*. FIPA has standardised one type of content expression language, FIPA-SL, in [FIPA00008]. Note other types of content expression have also been specified such as W3C-RDF and a constraint language but these did not mature into standard specifications.

[FIPA0008] defines SL in terms of a string expression grammar, defined to be a sub-grammar of the more general s-expression syntax. SL defines Content Expressions in terms of Action Expressions or Propositions. These in turn are represented as Well-Formed Formulas (wff) consisting of terms (Constant, Set, Sequence, Functional Term, Action Expression) and Constants (Numerical Constants, String, DateTime). A well-formed formula is constructed from an atomic formula by applying one of the construction operators or logical connectives operators, see Table 5. Different subsets of SL have been defined in [FIPA0008] depending on which operators are supported: SL0, SL1 and SL2. FIPA SL1 extends the minimal representational form of FIPA SL0 by adding Boolean connectives to represent propositional expressions, such as *not*, *and*, *or*. FIPA SL2 extends SL1 by adding construction, logic modal operators and the action operator feasible. SL2 allows first order predicate and modal logic, but is restricted to ensure that it must be decidable. Well-known effective algorithms exist that can derive whether or not an FIPA SL2 Wff is a logical consequence of a set of Wffs (for instance KSAT and Monadic). Note also that different CAs require the use of different SL subsets,

e.g., Query requires the use of a referential operator to assign a value for the results, defined in SL2 whereas Request requires use of the action operator done defined in SL0.

Operator Name	Operator Type	Example Wff	Description	SL
Negation	Boolean logic	(not <Wff>	The truth value of this expression is false if Wff is true. Otherwise it is true.	SL1
Conjunction	Boolean logic	and <Wff0> <Wff1>	This expression is true iff[4] well-formed formulae Wff0 and Wff1 are both true, otherwise it is false.	SL1
Disjunction.	Boolean logic	(or <Wff0> <Wff1>)	This expression is false iff well-formed formulae Wff0 and Wff1 are both false, otherwise it is true.	SL1
Implication	Predicate logic	(implies <Wff0> <Wff1>)	This expression is true if either Wff0 is false or alternatively if Wff0 is true and Wff1 is true. Otherwise it is false. The expression corresponds to the standard material implication connective Wff0 Wff1.	SL2
Equivalence	Predicate logic	equiv <Wff0> <Wff1>)	This expression is true if either Wff0 is true and Wff1 is true, or alternatively if Wff0 is false and Wff1 is false.	SL2

[4] If and only if.

			Otherwise it is false.	
Universal quantifier	Predicate logic Quantifier	forall <variable> <Wff>	The quantified expression is true if Wff is true for every value of value of the quantified variable.	SL2
Existential quantifier	Predicate logic Quantifier	exists <variable> <Wff>	The quantified expression is true if there is at least one value for the variable for which Wff is true.	SL2
Belief	Modal (BDI) Logic	B <agent> <expression>	It is true that agent believes that expression is true.	SL2
Uncertainty	Modal (BDI) Logic	U <agent> <expression>	It is true that agent is uncertain of the truth of expression. Agent neither believes expression nor its negation, but believes that expression is more likely to be true than its negation.	SL2
Intention	Modal (BDI) Logic	I <agent> <expression>	It is true that agent intends that expression becomes true and will plan to bring it about.	SL2
Persistent goal	Modal (BDI) Logic	PG <agent> <expression>	It is true that agent holds a persistent goal that expression becomes true, but will not necessarily plan to bring it about.	SL2
Feasible	Temporal	(feasible <ActionExpression> <Wff>	It is true that ActionExpression (or, equivalently, some event) can take place and just afterwards Wff will be true.	SL2
		(feasible <ActionExpression>	Same as (feasible <ActionExpression> true).	
Done	Temporal	(done <ActionExpression> <Wff>)	It is true that ActionExpression (or, equivalently, some event) has just taken place and just before that Wff was true.	SL0
		done <ActionExpression>	Same as (done <ActionExpression> true).	
Iota	Referential	(iota x (P x))	The expression means “the x such that P [is true] of x”. The iota operator introduces a scope for the given expression (which denotes a term), in which the given identifier, which would otherwise be free, is defined.	SL2
Any	Referential	(any <term> <formula>)	The any operator is used to denote any object that satisfies the proposition represented by formula.	SL2

All	Referential	(all <term> <formula>)	The all operator is used to denote the set of all objects that satisfy the proposition represented by formula.	SL2

Table 5. Content Expression operators defined in the FIPA Semantic language specification, FIPA00008. Different subsets of SL such as SL0, Sl1, SL2 use different sets of these operators. Wff represents a well-formed formula.

As mentioned earlier, FIPA separates out support for expressing operations on the content (defined in the content expression language) from support for referencing domain specific concepts, see Figure 8. Domain specific constant and variable terms in the content expressions can be explicitly referenced from application specific Ontologies. FIPA does itself specify any particular representation for the Ontology, e.g., W3C-OWL, or any particular level of expressivity for Ontologies. Ontologies ranging from so called weak Ontologies such as taxonomies to strong Ontologies that also support logical inferencing can be used. Thus, agents can not only share meta-concepts and concepts they can also share that may not understand concepts that they receive or to explain actions on concepts have failed.

```
(cfp
 :sender (agent-identifier : j)
 :receiver (set (agent-identifier : i))
 :content
   "((action (agent-identifier : i)
    (sell plum 50))
   (any ?x (and (= (price plum) ?x) (< ?x 10))))"
:ontology fruit-market
:language fipa-sl)
```

Figure 8. Agent j sends a call for proposals communicative act to ask i to submit its proposal to sell 50 boxes of plums. The generic content expression meta-concepts defined in the (content expression) language, given in bold. The (fruit-market) domain specific concepts defined in the domain Ontology (language), given in italics.

2.3 Process-oriented / Interaction Model

In a service-oriented distributed computing model, messages are not exchanged in isolation but are part of an interaction sequences: partially ordered plans or sequences of messages where partial ordering indicates that there are different choices of parts of the sequence.

FIPA agent interaction assumes the interaction assumes play the role of peers rather than use client server interaction. Typically one peer agents initiates interaction and then other peer agents act as participants, cooperating with the initiation to conclude the interaction. For many of these interactions, they can be cancelled by the initiator. Participants are also free to refuse interactions from initiators and to indicate that they don't understand messages or that they fail to carry out an interaction they have initially agreed to.

The simplest type of interaction is a two-way, two party, pull-type interaction when a sender has a priory knowledge of the ID, address and service interface of a receiver server. Then the interaction is a request by a sender followed by a reply from the receiver. For multi-party interaction or when sender needs to dynamically bind to a service interface, more complex interaction is needed. More complex

interaction includes service invocation via mediators, work-flows, information exchange and task delegation. FIPA has standardised several interaction protocols in [FIPA0023], [FIPA0026], [FIPA0037], [FIPA0028], [FIPA0029], [FIPA0030], [FIPA0001], [FIPA0035] and [FIPA0036].

Short descriptions of these protocols are as follows:

FIPA-Brokering Protocol: a broker is an agent which offers a set of communication facilitation services to other agents using some knowledge about the requirements and capabilities of those agents. A typical example of brokering is one in which an agent can request a broker to find one or more agents who can answer a query. The broker then determines a set of appropriate agents to which to forward the query, sends the query to those agents, and relays their answers back to the original requestor. The use of such brokerage agents can significantly simplify the task of interaction with agents in a multi-agent system. Brokering agents also enable a system to have adaptability and robustness for dynamic situations, scalability, and security control at the brokering agent.

FIPA-Recruiting Protocol: is similar to the brokering protocols but in the case of recruiting, the answers from the selected target agents go directly back to the original requestor or some designated receivers. The use of such brokerage agents can significantly simplify the task of interaction with agents in a multi-agent system. Brokering agents also enable a system to have adaptability and robustness for dynamic situations, scalability, and security control at the brokering agent.

FIPA-Subscribe-Protocol: an agent requests to be notified whenever a condition specified in the subscription message becomes true.

FIPA-ContractNet-Protocol is a minor modification of the original contract net protocol proposed of the widely used Contract Net Protocol, originally developed by Smith and Davis, in that it adds rejection and confirmation communicative acts. In the contract net protocol, one agent takes the role of manager. The manager wishes to have some task performed by one or more other agents, and further wishes to optimize a function that characterizes the task. This characteristic is commonly expressed as the price, in some domain specific way, but could also be soonest time to completion, fair distribution of tasks, etc.

The FIPA-Iterated-ContractNet-Protocol is an extension of the basic contract net. It differs from the basic version of the contract net by allowing multi-round iterative bidding.

FIPA-English-Auction-Protocol: the auctioneer seeks to find the market price of a good by initially proposing a price below that of the supposed market value, and then gradually raising the price. Each time the price is announced, the auctioneer waits to see if any buyers will signal their willingness to pay the proposed price. As soon as one buyer indicates that it will accept the price, the auctioneer issues a new call for bids with an incremented price. The auction continues until no buyers are prepared to pay the proposed price, at which point the auction ends. If the last price that was accepted by a buyer exceeds the auctioneer's (privately known) reservation price, the good is sold to that buyer for the agreed price. If the last accepted price is less than the reservation price, the good is not sold.

FIPA-Dutch-Auction-Protocol: the auctioneer attempts to find the market price for a good by starting bidding at a price much higher than the expected market value, then progressively reducing the price until one of the buyers accepts the price. The rate of reduction of the price is up to the auctioneer, and the auctioneer usually has a reserve price below which it will not go. If the auction reduces the price to the reserve price with no buyers, the auction terminates.

FIPA-Query-Protocol: the receiving agent is requested to perform some kind of inform action. Requesting to inform is a query, and there are two query-acts: query-if and query-ref. Either act may be used to initiate this protocol. If the protocol is initiated by a query-if act, it the responder will plan to return the answer to the query with a normal inform act. If initiated by query-ref, it will instead be an

inform-ref that is planned. Note that, since inform-ref is a macro act, it will be an inform act that is carried out by the responder.

FIPA-Propose-Protocol: an initiator agent proposes to the receiving agents that the initiator will do actions described in the proposed communicative act when the receiving agents accept this proposal. An example of the use of such a protocol is to propose a resolution to a group as a preamble to voting.

FIPA-Request-Protocol: allows one agent to request another to perform some action, and the receiving agent to perform the action or reply, in some way, that it cannot. Note also that a participant can give an optional quick agree as an acknowledge to a sender's CA that it will carry out later, perhaps because it is lengthy, or it can give the response and miss out the agree, perhaps because it is more expedient to do so.

FIPA-Request-When-Protocol: is simply an expression of the full intended meaning of the request-when action. The requesting agent uses the request-when action to seek from the requested agent that it performs some action in the future once a given precondition becomes true. If the requested agent understands the request and does not refuse, it will wait until the precondition occurs then perform the action, after which it will notify the requester that the action has been performed. Note that this protocol is somewhat redundant in the case that the action requested involves notifying the requesting agent anyway. If it subsequently becomes impossible for the requested agent to perform the action, it will send a refuse request to the original requestor.

These interaction protocols are characterised with respect to task vs. information sharing, push vs. pull and one to one vs. one-to-many receivers in Table 4. Unlike models of the individual communicative acts, the interaction models have a temporal dimension that can support branching-time, task duration and parallel execution.

Interaction Protocol	Task / info-sharing	Push / Pull	1-1 / 1-m receivers	Other features
Request	Task	Pull	1-1	Cancellable (by initiator)
Request-when(ever)	Task	Push	1-1	Cancellable
Query	Info.	Pull	1-1	Cancellable
Contract-Net/Iterated CN	Task	Push	1-m	Cancellable, iterated version is a multi-round IP
English / Dutch Auction	Info	Pull	1-m	Cancellable
Broker	Info	Pull	1-m	Cancellable
Recruit	Task	Pull	1-m	Cancellable
Subscribe	Info	Push	1-1	Not cancellable
Propose	Task	Pull	1-1	Not cancellable

Table 6. FIPA Interaction Protocols characterised with respect to task vs. information sharing, push vs. pull and one to one vs. one-to-many receivers.

Earlier interactions were represented as finite state-machine automata type graphical models but this lacked a standardised notion and expressivity of AUML.

The FIPA Request Interaction Protocol (IP) is an example of a FIPA IP that allows one agent to request another to perform some action. The representation of this protocol is given in Figure 9 which is based on extensions to UML 1.x. [Odell2001]. This protocol is identified by the token `fipa-request` as the value of the `protocol` parameter of the ACL message.

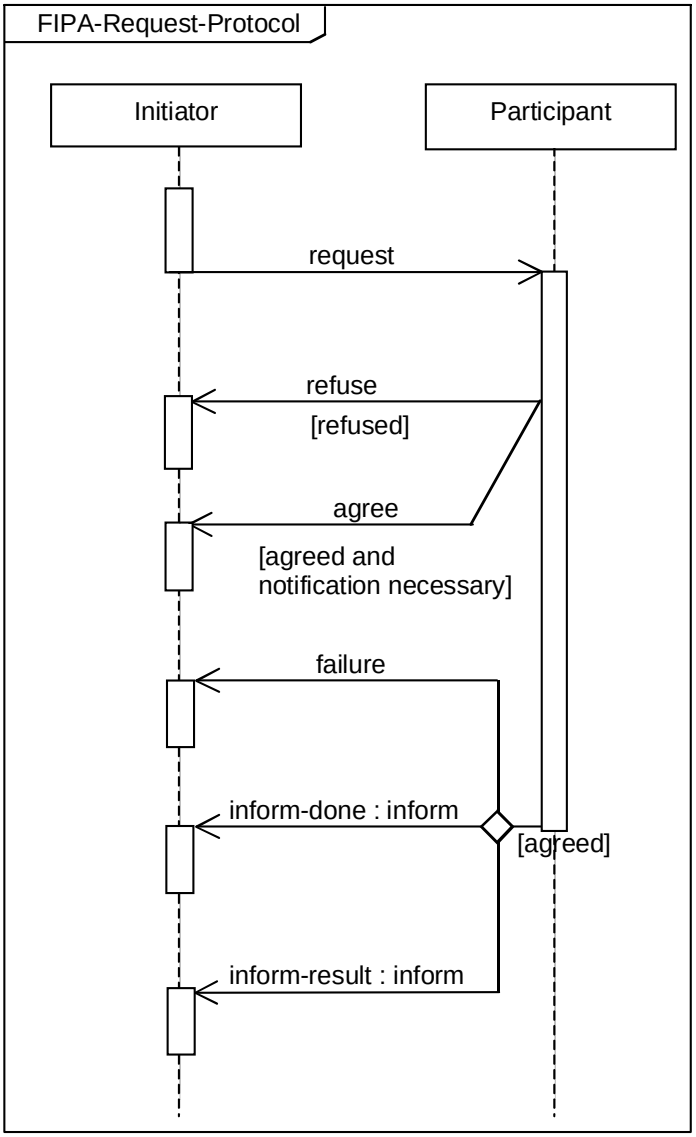


Figure 9: FIPA Request Interaction Protocol

The FIPA Request Interaction Protocol (IP) allows one agent to request another to perform some action. The Participant processes the request and makes a decision whether to accept or refuse the request. If a refuse decision is made, then “refused” becomes true and the Participant communicates a `refuse`. Otherwise, “agreed” becomes true.

If conditions indicate that an explicit agreement is required (that is, “notification necessary” is true), then the Participant communicates an `agree`. The `agree` may be optional depending on circumstances, for example, if the requested action is very quick and can happen before a time specified in the `reply-by` parameter. Once the request has been agreed upon, then the Participant must communicate either:

- A `failure` if it fails in its attempt to fill the request,
- An `inform-done` if it successfully completes the request and only wishes to indicate that it is done, or,
- An `inform-result` if it wishes to indicate both that it is done and notify the initiator of the results.

Any interaction using this interaction protocol is identified by a globally unique, non-null `conversation-id` parameter, assigned by the Initiator and set in the ACL message structure. The agents involved in the interaction must tag all of its ACL messages with this conversation identifier. This enables each agent to manage its communication strategies and activities, for example, it allows an agent to identify individual conversations and to reason across historical records of conversations.

At *any* point in the IP, the receiver of a communication can inform the sender that it did not understand what was communicated. This is accomplished by returning a `not-understood` message. As such, Figure 9 does not depict a `not-understood` communication as it can occur at any point in the IP. The communication of a `not-understood` within an interaction protocol may terminate the entire IP and termination of the interaction may imply that any commitments made during the interaction are null and void.

At any point in the IP, the initiator of the IP may cancel the interaction protocol by initiating the meta-protocol shown in Figure 10. The `conversation-id` parameter of the cancel interaction is identical to the `conversation-id` parameter of the interaction that the Initiator intends to cancel. The semantics of cancel should roughly be interpreted as meaning that the initiator is no longer interested in continuing the interaction and that it should be terminated in a manner acceptable to both the Initiator and the Participant. The Participant either informs the Initiator that the interaction is done using an `inform-done` or indicates the failure of the cancellation using a `failure`.

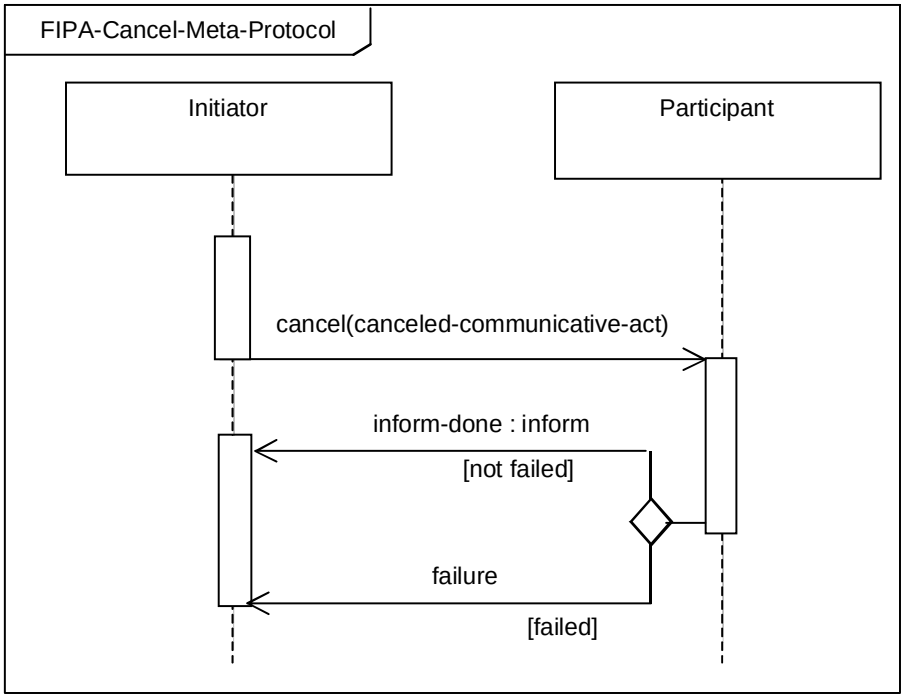


Figure 10: FIPA Cancel Meta-Protocol

2.4 Service Model

In the FIPA Model, services represent components of the architecture that generate, consume and transform ACL messages. In the general case such service components are not mandated to be agents themselves although they could be.

2.4.1 Abstract Architecture Model

The primary focus of this abstract architecture [FIPA0001] is to specify abstract service architecture for message exchange between agents which may be using different messaging transports, different Agent Communication Languages and different content languages. This requires numerous points of potential interoperability. The scope of this architecture includes:

- Message transport interoperability.
- Supporting various forms of ACL representations.
- Supporting various forms of content language.
- Supporting multiple directory services representations.

Element	Description	Mandatory Optional
Action-status	A status indication delivered by a service showing the success or failure of an action.	Mandatory
Agent	A computational process that implements the	Mandatory

	autonomous, communicating functionality of an application.	
Agent-attributes	A set of properties associated with an **agent** by inclusion in its **directory-entry**.	Optional
Agent-name	An opaque, non-forgeable token that uniquely identifies an **agent**.	Mandatory
Agent-communication-language	A language with a precisely defined syntax semantics and pragmatics, which is the basis of communication between independently designed and developed **agents**.	Mandatory
Content	**Content** is that part of a communicative act that represents the domain dependent component of the communication.	Mandatory
Content-language	A language used to express the **content** of a communication between agents.	Mandatory
Directory-entry	A composite entity containing the **name**, **locator**, and **agent-attributes** of a **agent**	Mandatory
Directory-service	A **service** providing a shared information repository in which **directory-entries** may be stored and queried	Mandatory
Encoding-representation	A way of representing an abstract syntax in a particular concrete syntax. Examples of possible representations are XML, FIPA Strings, and serialized Java objects.	Mandatory
Envelope	That part of a **transport-message** containing information about how to send the message to the intended recipient(s). May also include additional information about the message encoding, encryption, etc.	Mandatory
Explanation	An encoding of the reason for a particular **action-status**.	Optional
Locator	A **locator** consists of the set of **transport-descriptions** used to communicate with an **agent**.	Mandatory
Message	A unit of communication between two agents. A **message** is expressed in an **agent-communication-language**, and encoded in an **encoding-representation**.	Mandatory
Encoding-transform-service	A **service** that transforms a **message** or **payload** from one **encoding-representation** to another.	Mandatory
Message-transport-service	A **service** that supports the sending and receiving of **transport-messages** between **agents**.	Mandatory

Ontology	A set of symbols together with an associated interpretation that may be shared by a community of **agents** or software. An ontology includes a vocabulary of symbols referring to objects in the subject domain, as well as symbols referring to relationships that may be evident in the domain.	Optional
Payload	A **message** encoded in a manner suitable for inclusion in a **transport-message**.	Mandatory
Service	A service provided for **agents** and other **services**.	Optional
Transport	A **transport** is a particular data delivery service supported by a given **message-transport-service**.	Mandatory
Transport-description	A transport-description is a self describing structure containing a transport-type, a transport-specific-address and zero or more transport-specific-properties.	Mandatory
Transport-message	The object conveyed from **agent** to **agent**. It contains the **transport-description** for the sender and receiver or receivers, together with a **payload** containing the **message**.	Mandatory
Transport-specific-property	A transport-specific-property is a property associated with a transport-type.	Optional
Transport-type	A **transport-type** describes the type of transport associated with a **transport-specific-address**.	Mandatory

Table 7: List of abstract elements defined in the Abstract Architecture Specification [FIPA00001]

The list of abstract elements defined in the Abstract Architecture Specification [FIPA00001] is given in Table 7. The elements of the abstract architecture are defined in a series of UML diagrams in [FIPA00001], an example is given in Figure 11.

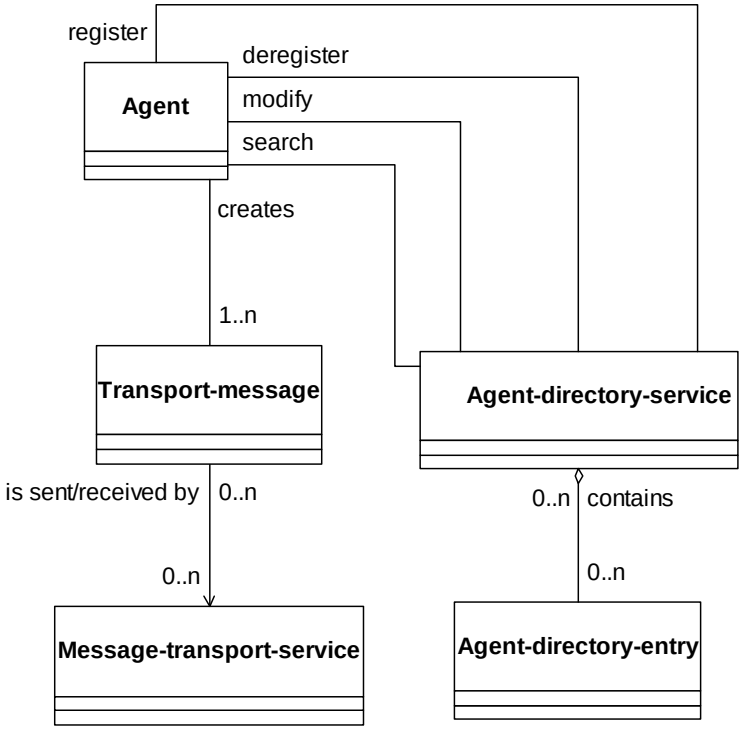


Figure 11: UML model of basic Agent Relationships

Some aspects of potential standardization are outside of the scope of this architecture. There are three different reasons why things are out of scope:

- The area cannot be described abstractly.
- The area is not *yet* ready for standardization, or there was not yet sufficient agreement about how to standardize it.
- The area is sufficiently specialized that it does not currently need standardization.

Some of the key areas that are not included in standardised architecture are: Agent lifecycle and management, Agent mobility, Domains, Conversational policy, Agent Identity.

The abstract architecture does not prohibit additional features – it merely addresses how interoperable features should be implemented. It is anticipated that over time some of these areas will be part of the interoperability of agent systems. The Abstract Architecture defines two support core services.

A *directory-service* is a shared information repository in which agents may publish their directory-entries and in which they may search for directory-entries of interest. A concrete instantiation of directory-service is a mandatory element of every concrete instantiation of the abstract architecture. The core actions supported by this service include: Register (service description), modify, deregister and search.

A *message-transport-service* is a service that supports the sending and receiving of transport-messages between agents. A concrete instantiation of message-transport-service is a mandatory element of every concrete instantiation of the abstract architecture. The core actions supported by this service include: Bind Transport(service to agent), Unbind Transport (service to agent) and Send Message. N.B. No receive message action is defined.

2.4.2 Reifying Abstract Architectures

It must be possible to create implementations that vary in some of these attributes, but which can still interoperate. An abstract architecture cannot be directly implemented, but instead the forms the basis for the development of concrete architectural specifications. Such specifications describe in precise detail how to construct an agent system, including the agents and the services that they rely upon, in terms of concrete software artefacts, such as programming languages, applications programming interfaces, network protocols, operating system services, and so forth.

In order for a concrete architectural specification to be FIPA compliant, it must have certain properties. First, the concrete architecture must include mechanisms for agent registration and agent discovery and inter-agent message transfer. These services must be explicitly described in terms of the corresponding elements of the FIPA abstract architecture. The definition of an abstract architectural element in terms of the concrete architecture is termed a *realization* of that element; more generally, a concrete architecture will be said to *realize* all or part of an abstraction.

The designer of the concrete architecture has considerable latitude in how he or she chooses to realize the abstract elements. If the concrete architecture provides only one encoding for messages, or only one transport protocol, the realization may simplify the programmatic view of the system. Conversely, a realization may include additional options or features that require the developer to handle both abstract and platform-specific elements, see Figure 12.

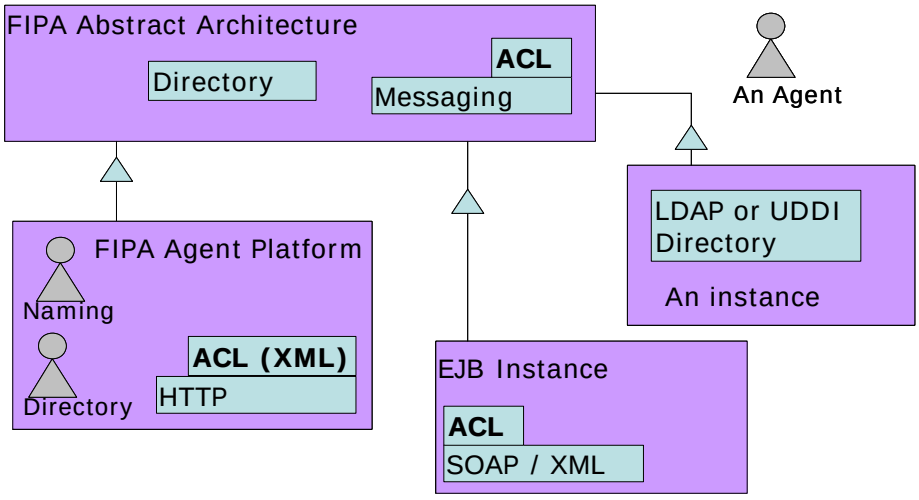


Figure 12. The FIPA architecture defines two basic services (directory and messaging). Three different concrete architectures that realise part of the abstract architecture are shown.

The abstract agent transport architecture supports the development of instantiations that use transports, encodings, and infrastructure elements appropriate to the application domain. To ensure that heterogeneity does not preclude interoperability, the developers of a concrete architecture must consider the modes of interoperability that are feasible with other instantiations. Where direct end-to-end interoperability is impossible, impractical or undesirable, it is important that consideration be given to the specification of gateways that can provide full or limited interoperability, see Figure 13. Such

gateways may relay messages between incompatible transports, may translate messages from one encoding to another, and may provide Quality of Service features supported by one party but not another.

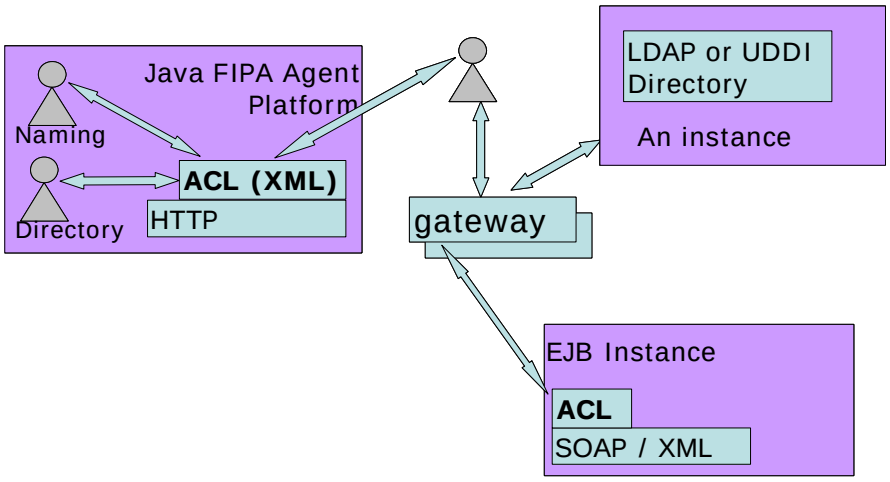


Figure 13: Abstract Architecture model of interoperability using gateways,

Gateways present one way of combining different architectural models. For example, a Web-Service Agent Gateway could be envisaged that maps FIPA-ACL messages to SOAP messages. However the mappings between different concrete messaging system types in gateways is considered out of scope for the abstract specification. Note also that this is likely to be a translation of the syntax and grammar and not the semantics so any semantics of the ACL messages would be lost in converting them to Web-service messages or even the Semantic Web. Other issues are that Web service descriptions may not easily relate to Agent descriptions. These issues are discussed more in Section 6.

2.4.3 Agent Management or Agent Platform Model

Agent Management (AM) or the agent platform (AP) is specified in [FIPA00023] and can be regarded as a concrete architecture that realises the abstract architecture [FIPA00001], see Figure 12. Some of the key differences between these models are that:

- [FIPA00023] specifies mandatory AP and AM elements whereas [FIPA00001] does not
- In [FIPA00001] a directory service is mandatory. An AM is defined in [FIPA00023] as the mandatory *white-page* type directory service. In addition, the AM entity supports agent management operations such as Suspend agent and Terminate agent.
- [FIPA00023] also specifies and additional *yellow-page* directory (facilitator) service as optional
- [FIPA00023] specifies a AM ontology for accessing directory and agent management services via agents but [FIPA00001] does not.

Agent management provides the normative framework within which FIPA agents exist and operate. It establishes the logical reference model for the creation, registration, location, communication, migration and retirement of agents. The entities contained in the reference model (see Figure 1) are logical capability sets (that is, services) and do not imply any physical configuration. Additionally, the implementation details of individual APs and agents are the design choices of the individual agent system developers.

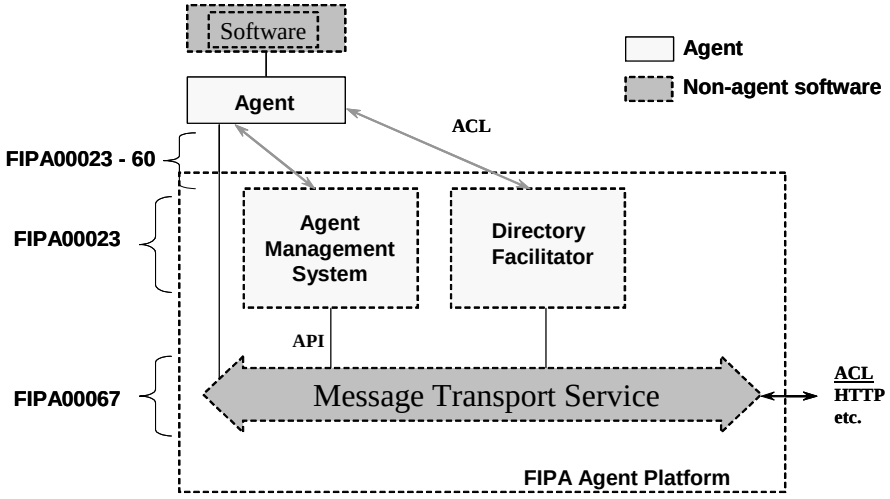


Figure 14: Agent Management Reference Model from [FIPA00023]. The numbers on the left refer to related FIPA specifications.

The agent management reference model consists of the following logical components, each representing a capability set (these can be combined in physical implementations of Agent platforms or APs):

An A*gent* is a computational process that implements the autonomous, communicating functionality of an application. Agents communicate using an Agent Communication Language. An Agent is the fundamental actor on an AP which combines one or more service capabilities, as published in a service description, into a unified and integrated execution model. An agent must have at least one owner, for example, based on organisational affiliation or human user ownership, and an agent must support at least one notion of identity. This notion of identity is the Agent Identifier (AID) that labels an agent so that it may be distinguished unambiguously within the Agent Universe. An agent may be registered at a number of transport addresses at which it can be contacted.

A *Directory Facilitator* (DF) is an optional component of the AP, but if it is present, it must be implemented as a DF service (see Section 4.1). The DF provides yellow pages services to other agents. Agents may register their services with the DF or query the DF to find out what services are offered by other agents. Multiple DFs may exist within an AP and may be federated. The DF is a reification of the Agent Directory Service in [FIPA00001].

An *Agent Management System* (AMS) is a mandatory component of the AP. The AMS exerts supervisory control over access to and use of the AP. Only one AMS will exist in a single AP. The AMS maintains a directory of AIDs which contain transport addresses (amongst other things) for agents registered with the AP. The AMS offers white pages services to other agents. Each agent must register with an AMS in order to get a valid AID. The AMS is a reification of the Agent Directory Service in [FIPA00001]. The AMS defines the core directory actions from [FIPA00001] such as `register, deregister, modify and search` In addition to the management functions exchanged between the AMS and agents on the AP, the AMS can instruct the underlying AP to perform the following agent management operations: Suspend agent, Terminate agent, Create agent, Resume agent execution, Invoke agent, Execute agent, and Resource management.

An *Message Transport Service* (MTS) is the default communication method between agents on different APs (see [FIPA00067]).

An *Agent Platform* (AP) provides the physical infrastructure in which agents can be deployed. The AP consists of the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS and MTS) and agents. The internal design of an AP is an issue for agent system developers and is not a subject of standardisation within FIPA. AP's and the agents which are native to those APs, either by creation directly within or migration to the AP, may use any proprietary method of inter-communication. It should be noted that the concept of an AP does not mean that all agents resident on an AP have to be co-located on the same host computer. FIPA envisages a variety of different APs from single processes containing lightweight agent threads, to fully distributed APs built around proprietary or open middleware standards. FIPA is concerned only with how communication is carried out between agents who are native to the AP and agents outside the AP. Agents are free to exchange messages directly by any means that they can support.

The agent platform services are defined using an agent management ontology in [fipa000067] as a set of frames that represent the classes of objects in the domain of discourse within the framework of this `fipa-agent-management` ontology. This ontology does not specify any specific positional order to encode the parameters of the objects. Therefore, it is required to encode objects in SL by specifying both the parameter name and the parameter value (see Section 3.6 of [FIPA00008]). An example of part of this agent management ontology that specifies the agent identifier concept is given in Table 8.

Frame	agent-identifier			
Ontology	fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
name	The symbolic name of the agent.	Mandatory	word	df@hap_name ams@hap_name
addresses	A sequence of ordered transport addresses where the agent can be contacted. The order implies a preference relation of the agent to receive messages over that address.	Optional	Sequence of `url`	
resolvers	A sequence of ordered AIDs where name resolution services for the agent can be contacted. The order in the sequence implies a preference in the list of resolvers.	Optional	Sequence of agent-identifier	

Table 8: example of part of the FIPA agent management Ontology defined in [FIPA00067] using a frame notion

2.5 Agent Development Methodology

FIPA did not standardise a development methodology for creating and maintaining agents, This is discussed more in a later section 4 and or 6.

2.6 Specification Representations

Generally, MAS systems and distributed systems in general are more popularly implemented using procedural programming API abstractions such as currently J2SE and .NET etc. However declarative approaches are found to more flexible and able to capture the requirements for the design and seem to

have a longer shelf-life in practice, examples of declarative approaches include SQL and XML. Declarative specifications can be specified using a more abstract notation and context-free grammars such as Backus Naur Form (BNF) or in a more concrete computational format such as XML. To date, FIPA has favoured using BNF where possible has the notation for specifying most of the communication protocol sub-layers, see Table 9.

	Abstract Spec.	Concrete spec. encoding	Notes
Architecture	Semantics, Graphical, UML		
Agent Management	Semantics, Frame-based Ontology	FIPA-SL1	
Application Sub-layer 1 (Transport)	External	HTTP/ IIOP	FIPA specifies constrainsts for transports to be reliable, ordered, asynchronous mode
Application Sub-layer 2(Encoding)	Syntax, BNF	XML, String, Bit	
Application Sub-layer 3(Messaging)	Syntax, BNF	XML, String or Bit	
Application Sub-layer 4 (Ontology)	External	Any	
Application Sub-layer 5 (Content expression)	Syntax, BNF	FIPS-SL2	There are 3 sub-sets: these differ by the range of operators they support.
Application Sub-layer 6 (Communicative Act or CA)	Semantics, 1st order modal logic (FIPA-SL)		
Application Sub-layer 7 (Interaction Protocol or IP)	Semantics, Graphical, A-UML		

Table 9: Abstract and concrete specification representations for the different communication protocol levels.

For sub-layers 6 and 7, FIPA wanted to specify semantics and so use a context-sensitive grammar syntax is used. To use these specifications, in practice requires the use of software parsers that parser the grammars. If the concrete encodings are used, then there are some well-known standard parsers available. However, parsers for FIPA-SL are less well-known. FIPA has used multiple representations to model and specify different aspects of MAS organisation, interaction or interfaces and behaviour.

The definition or semantics of CA are modelled in terms of a type of formal or modal logic FIPA-SL [FIPA0008]

IPs are defined using AUML, e.g., [FIPA000?]

Architecture: the abstract architecture for a FIPA MAS is modelled using UML [FIPA0001]. The more concrete agent platform architecture is not modelled using UML but using a non-standard graphical notation.

Services are modelled using domain specific framed based Ontology, e.g., FIPA management ontology [FIPA0023], device ontology [FIPA0009] and the quality of service [FIPA0094].

It is probably true that no single notation would be suitable to specify all aspects of FIPA [MAS] systems. In addition, it is advantageous that that representation used is: abstract, expressive to supports the different agent MAS properties, a computational form, supports model-checking, support software engineering. These have various degrees of formalism (define this), degrees of computation models (define), design implementation or computable

3 Deployed FIPA MAS Systems

3.1 Introduction

In this section an analysis of the design and the design choices made by FIPA is presented. There is a wealth of research papers in the wider MAS community that have investigated some of these issues in detail. In this section the idiosyncrasies of the theoretical model are discussed. Whereas in the next section, deployment gap - the gap between the theoretical design model and the operational or computation implementation model, issues are discussed.

3.2 Some Experiences in using the Specifications

See http://www.fipa.org/resources/projects.html but this is not up to date. Whilst there are many reported uses of the FIPA projects, we highlight some experiences where researchers and developers have published some greater insightful and critical experience in using the FIPA specifications. The main phases of application development can be summarised: agent roles and responsibilities within the scenario; Common Ontology Development and Application Specific interaction Protocols.

3.2.1 FACTS Project

FACTS is an EU-sponsored collaborative project, part of the ACTS programme, project number AC317 which ran 1998-2000, whose objective was to validate and drive the FIPA specifications. Three main domain applications have been chosen in order to test the standard in real-life scenarios: personal travel market (PTM), electronic trading and audio-visual entertainment and broadcasting (AVEB).

Experience at developing the PTM application are reported by Núñez-Suárez et al (2000) using a Multi Agent System based upon the existing travel industry that incorporates electronic equivalents of travel agents and service providers and an electronic assistant acting on behalf of the user. Three different types of agent involved in the scenario. The Personal Travel Assistant (PTA) resides upon on type of FIPA agent platform called the Agent Services Layer (ASL) platform whereas the Travel Broker Agent (TBA) and Travel Service Agents (TSAs) reside upon the JADE agent platform. Their main findings were that:

The FIPA model reduced the amount of work required to attain application level interoperability, shifting the focus away from infrastructure and interfaces and towards application behaviour.

The openness of FIPA supports the ability to integrate very disparate components was demonstrated.

The use of technologies such as Interaction Protocols, Agent Communication Languages and Ontology may be common within the agent community, but they are certainly not common within the mainstream software development industry This highlighted an urgent need for agent system development tools which hide complexity from the developer yet provide her with the ability to model the knowledge domain and to develop agent based systems using more common methodologies

In practice, the informative definitions of the Communicative Acts (CA) are intuitive and comprehensive enough for use in most cases.

There was a low-level bootstrapping problem associated with the use of the IIOP transport in that object references to key services such as the agent platform ACL transport needed to be known a priori This is outside the specifications and needs to be solved in an implementation specific way.

Experiences in developing the FACTS AVEB application have been reported by Charlton et al (2000). The AVEB application is primarily motivated by the consideration that TV programs on offer will soon exceed the monitoring capabilities of the typical user, users will be in need of a more sophisticated support in the selection of interesting programs, as well as in the negotiation of the conditions at which

programs and services are purchased. A MAS solution was developed to support this requirement. Their reported findings can be summarised as:

ACL semantics is based on this mental agency which represents the private context of an agent, applications tend not to implement these semantics as computational modes of these are complex to build and operate, for example there may be a limited distinction between intentions and desires in implementation.

ACL semantics based upon a social agency or organisational model can be used to overcome the problems of mental agency model. It will need to separate out domain independent concerns from domain specific concerns. Currently, in many systems, the social agency is more implicit and related to the ad hoc explicit interaction a MAS supports. Note since this time, FIPA has specified a set of predefined interaction protocols with an associated social agency model.

Error handling is a concern, if agents are autonomous, they can delay responding, causing the sender to be uncertain what has happened. Since this report, FIPA has added time-out fields to the protocol headers. Also there is a limited support to debug and monitor platforms. Since this time, Agent platforms have built added debugging tool support.

3.2.2 MARINER Project

The EU-funded MARINER project is concerned with developing a FIPA compliant agent system for load control in intelligent networks. In MARINER scenario 1, quantifier agents are concerned with periodically computing the cost of an IN service to another type of allocator network element agents. Pitt and Mamdani [1999] report their experiences of using the FIPA specifications within the MARINER project. Their primary criticisms of the FIPA ACL standards are firstly, that it is not clear whether the ACL semantics should be interpreted as an informative specification providing guidance to developers, or as a normative specification providing conditions that the agents themselves are responsible for satisfying. Secondly, that any communication semantics will vary depending on the context and therefore be difficult to standardise. They argue that interaction protocols or conversations should arguably be the focal point for standardisation of the ACL as this can provide the context for communication. They also the following design guidance:

- Design for error: to handle ACL exceptions that are expected to arise because of the autonomy, concurrency, non-determinism, asynchrony, and possible non-termination of distributed communication to improve robustness.
- Separation of concerns: cleanly separate the communication context semantics from the content semantics but yet allow the two to be interlinked when needed.
- Classify speech acts and where possible evolve new speech acts from old ones. Even if the semantics of the individual CAs are not so useful for interoperability, they provide a useful structured framework for discussing design issues of CAs, differentiating CAs and deriving new CAs and CA frameworks in an evolutionary way.
- Concentrate on conversations.

3.2.3 Agentcities Project

The main objective of the Agentcities.rtd of the EU FP5 funded project was to create a global open system to provide the conditions in which to test FIPA based agents, services and other technologies, such as delegation, coordination, modelling of dynamics and, in particular, communication based on formal semantics (Willmot, 2002). The main achievements were (Agentcitie.RTD, 2002):

Agentcities Network Architecture: defined an architectural model for the development and deployment of large-scale networks of autonomous service components. The network architecture is divided into

three levels: A high level description of the key elements and structures to be found in the open service networks; Reifications of these high level models to a number of technology sets and particular interface definitions; Specification of deployed instances of networks including operational policies, structural descriptions and bindings to particular service instances

Service Interoperability: Agentcities.RTD provided a reference model for semantic interoperability in open environments into which different technologies can be inserted and which can be reused in different technology environments such as agent, systems, Grids and e-Business Frameworks and so forth).

Service Composition: Agentcities.RTD developed a framework for service composition in an open environment that was instantiated using the FIPA standards and a number of open-source & proprietary software toolkits into functioning use cases.

Test suite for FIPA-compliant Agent Platforms: a tool was created online to test the main interoperability features of FIPA platforms.

Ontologies and services Agentcities:RTD defined Ontologies and trialled services in the following domains: Auction Houses, Banking, shows, Ticketing and Market Places, Security, Restaurant, Cinema, Hotels, personal information management, Geographic Information, Transportation and Evening Organizer and Event Organizer

Live Network: An Agentcities testbed network was actively used by a wide range of organisations and this usage is supported by a range of network services which enable: service advertisement/discovery, identity management, communication as well as basic management.

There was also an associated Agentcites.Net project that allowed even more services and agents to be hooked up and tested.

3.3 FIPA Tools and software APIs

Application developers rather than having to develop their own software implementations of the FIPA specifications typically use agent toolkits and layer their application software on top. This eases development and the amount of testing assuming the agent toolkits undergo some form of evaluation. During the lifetime of FIPA, several tens of FIPA agent toolkits have been developed that have implemented sets of FIPA specifications. Here we mention some of the main open source initiatives: JADE (Bellifemine, 1999), FIPA-OS (Poslad, 2000) and ZEUS (Nwana, 1999) that were used in the FIPA interoperability tests. In addition, a JCP or Java Community process developed JAS, the Java Agent Service, JSR87, reference API for the FIPA abstract architecture specification that has been implemented in the KAoS agent toolkit (Bradshaw, 2004). These toolkits typically support implement and provide APIs and tools as follows:

- APIs and implementations of codecs to parse FIPA ACL messages in accordance with the FIPA CA library and other related ACL specifications a
- APIs to support for agent management as defined in [FIPA0023] rather than support for the abstract agent architecture.
- Different types of Agent templates for producing agents which can then communicate with each other using different facilities;
- Multi-layered support for agent communication;
- Message and conversation management to support interaction protocols

- Dynamic platform configuration to support multiple agent platforms, multiple types of persistence (enabling integration with legacy persistence software) and multiple encodings.
- Abstract interfaces and software design patterns
- Diagnostics and visualization tools

3.3.1 Others

3.3.2 How toolkits deal with the ACL semantics and other theoretical properties

Software APIs and parsers for computational models of the specifications can be readily developed to exchange ACL messages and the associated sub-layer protocols at a syntactical level. One of the key technical challenges is how to implement software that can handle the ACL messages at a semantic level. Putting aside for the moment, the concerns about the validity and verifiability of the CA semantics, some researchers (Louis and Martinez, 2004) have reported their efforts to develop a computation framework to handle the CA semantics that has been implemented as an add-on to JADE. This framework has four elements: an activity loop to send and receive messages, reflexive operations to access and update their beliefs, a Semantic Representation (SR) to process messages with beliefs expressed in FIPA-SL and Semantic Interpretation Principles (SIP) to produce and consume SRs. They consider 3 main ways to implement their model: using a dedicated inference engine for modal logic but these tend to be not publicly available; to use a general purpose rule engine to help process messages but this requires some mapping the semantic model to the rule engine rules and finally to develop their a proprietary ad hoc rule engine. Note that their framework focuses on belief transfer but not intention transfer although they claim that the latter can be easily added.

An alternate approach is that most agent applications developers choose in practice is not to use and implement the CA semantics but to use some informal semantics based upon the interaction context. The intentional semantics of an agent that is the meaning of the CA will be determined by the state of an agent is in, in relation to a priori agreed interaction protocol (IP), and the type of the next received CA or sent CA. In (Pitt and Bellifemine, 1999b), a JADE implementation of such IP driven intentional semantics is described.

3.4 Interoperability testing and FIPA compliance

It is not so useful in practice when dealing with a protocol suite such as FIPA or the IETF TCP/IP protocols to speak of compliance in the sense to the whole set of protocols but rather to be concerned with compliance to specific protocols. Certain individual protocols may be core such as using an FIPA ACL syntax where as other protocols are domain specific such as using an interaction protocol or even the use of the formal semantics for the set of FIPA CA, although the latter point is somewhat controversial.

Several authors, such as Wooldridge (2000), Pitt (1999), Louis (2004) have pointed that conformance to the specification only requires the sender to respect the feasibility condition in order to send the message but does not require the recipient of a message to respect the rational effect part of the CA semantics. There are other challenges to do with the intentional model mentioned earlier. There is also the further issues of the computation tractability when verifying open service interaction, where a system that interacts with its environment and whose behaviour depends on this interaction, an issue not considered in the above references. These issues make computational models and conformance testing of MAS to the formal semantics, challenging in practice.

The concern of application developers and users is less on whether or not the application communication can be formally verified against some underlying theoretic model but more on whether

they can apply a model and that model performs as expected and that the computation implementations of the models, not the theoretical models, can be evaluated to pass a series of tests. The construction of these ad hoc tests and the specification of the test conditions and test properties is application specific.

There are several ways implementations of the specifications are tested in practice. The specifications are tested as part of the experimental phase in progressing a specification from preliminary via experimental to standard. Specifications are testing as part of good development practice within specific projects that used these specifications. In addition, several specific tests of the specifications were carried out as follows at selected FIPA meetings such as the London FIPA meeting in 2001 where the interoperability between the JADE, FIPA-OS and ZEUS three platforms was tested. The specifications tested included: Agent Management (FIPA0023), Message Transport Service or MTS (FIPA0067), MTP for IIOP (FIPA0075), MTP for HTTP (FIPA0084), Agent Communication Library (FIPA0037), String ACL Encoding (FIPA0070), Bit-efficient ACL Encoding (FIPA0069), FIPA-SLO part of the SL content language ((FIPA0008), FIPA-Request Interaction Protocol (FIPA0026) and FIPA-Query Interaction Protocol (FIPA0027). The results of the testing were two-fold. It 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 a 1999 FIPA meeting. FIPA specifications were considered fairly thorough, in that, only minor changes were necessary to enable the “core functions” to be tested.

4 Features and Constraints of the Models

4.1 ACL Model features

The FIPA-ACL model is flexible in the sense that a more abstract model and notation is used for a specification of parts that are then grounded or reified using particular concrete models. This is has the advantage over grounding a model using the particular technology of the day in the it makes the specifications less fragile but this can introduce increased interoperability problems.

There are alternatives for different parts of the layered model given in Figure 3.

- FIPA-ACL Syntax
- Transport protocol: IIOP, HTTP, bit-efficient
- Content Language: SL, SL-1, SL-2, SL3. Talk about other candidate ones that were considered but never standardised such as W3C RDF and a constraint language,
- Interaction protocol instances: FIPA agents are also allowed to interact without using a standard FIPA IP
- CA instances
- Domain Ontologies

The following components of the ACL models have no alternatives

- Semantics of individual CA must use the BDI semantics

Technology specific versus technology neutral model: The advantages of having a single protocol choice at each sub-layer may make boot-strapping and interoperability easier but there may be no optimum choice across multiple domains, e.g., at the time the first set of FIPA specifications were produced in 1997, one of the dominant distributed computing models was the OMG CORBA or Common Object Request Broker Architecture and its main transport protocol was the IIOP protocol.

However this protocol was difficult to use in low bandwidth wireless environments and in high-throughput transaction processing environments.

Semantics and conceptualisation for the whole ACL model: There is no holistic conceptual structure to link concepts in one sub-layer protocol of the FIPA-ACL to be interlinked to those in another, e.g., to make it easier to check that the request CA contains a sub-action , that a query CA contains a free variable to hold the results of an information query. This lack of holistic structure is further complicated by the lack of a common representation for the different conceptualisations at different communication sub-layers. Cranfield et al (2005) have proposed an Ontology that can be used to interlink the concepts in different ACL sub-layers in order to have an on-line representation of the structural semantics of the whole ACL

4.2 CA Model Features

4.2.1 Use of BDI semantics for CA

The CA semantics have been formally defined in terms of a modal logic, See section 2.2.1. This section consider the semantics for the communication itself as opposed to the semantics of the specific content being communicated such as knowledge exchange or task sharing. There are several advantages for using such a formal model for communication such as greater expressiveness. For example, the ability to differentiate between that an agent knowing some specific action or set of actions that achieves a goal versus that agent may believe the goal can be achieved without knowing a specific set of actions to help achieve it (Louis and Martinez, 2004).

FIPA CA semantics as viewed by the sender's mental attitude or BDI Model. Although there are principles for transferring the mental attitude in terms of belief and intention from the sender to the receiver inherent in the model, the actual interpretation of the sender's intentional effect in the receiving agent is considered to be relative to each agent and customisable by each agent. The process of actually interpreting the sender's intent via messages at the receiver, such as using some FIPA specific BDI rule engine, is not specified by FIPA although this has been proposed and discussed several times in FIPA meetings.

Meaning of CA varies dependent on the context: any interaction between intelligent entities such as using an intentional stance to articulate the meaning of communication between agents is likely to be heavily context dependent. An attempt to standardise communication is likely to work for some, but not all, contexts. Examples of this include: an agent that needs to be not strictly sincere, an agent that wants to confirm a previous arrangement, an agent that is continuously updating information; an agent that wants to withdraw specific information to selected agents rather than disconfirm information (Pitt & Mamdani, 1999). Reed et al (2002) also present a similar argument not to fix the semantics of individual CA absolutely but to specify a framework that allows some dimensions of a CA to vary and be fixed at run-time. They propose an approach they call semantic fixing in which the preconditions of a CA, the postconditions of a CA and CA beleifs can be varied and controlled depending upon the position an agent takes and its freedom to act in relation to norms that are established during contract type interaction. Additional sub-interactions are however needed, that may involve multiple rounds and voting in order to agree on the semantics of the CA between parties. This however can significantly increase the computation overhead to the interaction and the approach proposed to fix the semantics may not be universal.

Agents act sincerely. The assumption in the interaction is that agents act sincerely – they always speak the truth and believe each other. Agents cannot seek to lie about their beliefs that they are communicated as being true when the sender knows they are indeed false. This can happen in many types of ecommerce and trade. This assumption is used in many MAS models as it is easier to design cooperative interaction when this is assumed to be true. It has been argued that if FIPA agents use the

BDI semantics they are too limited for use in ecommerce, however, note that the FIPA agents often use other semantics such as IP protocol semantics

*Other criticisms and limitations of the BDI mode*l: There are a number of different variations of BDI theories in terms such as the number and choice of modalities, e.g., intention being entailed as a modality or defined as a first class modality, BDI models have incomplete axiomisations and can be computational complex or even intractable. BDI model focuses on private belief and intention transfer between individuals rather and don't take into account third party or societal interaction and associated constraints. BDI models seldom focus on pragmatic issues such as belief and intention managements in an open system that make the model computationally complex or even intractable for example, how beliefs are established, how to deal with inconsistent, partial, probalistic, conflicting, cyclic and precedence of beliefs.

4.2.2 Use of alternative (to BDI) semantics for FIPA-ACL

Third-party semantics based upon social commitments. As the FIPA ACL semantics can be considered to focus on transferring the sender's mental attitude to one or more receivers but models of society or third parties are not considered (Singh, 1998). In this model agents play different roles within society and these roles define associated social commitments that constrain how agents playing a role must act and communicate. For example, one agent can allocate a task to other agents, which is consistent with its mental attitude but the task may not be allowed because of organisational constraints, the agent does not have the authority to carry out the task even although it has the capability to do so.

Use of IP Semantics: The FIPA Interaction Protocol model makes a rudimentary attempt at a social model in the sense that the interaction is related to the organisational roles of the interacting parties and the semantic of each CA in an IP is interpreted within the context of the IP.

Contract programming model semantics: Agent communication can be specified without being formally specified. There are many proprietary MAS that interact in closed systems in this way. For example, KQML or Knowledge Query Meta Language) model uses a type of programming by contract model to specify its semantics in terms of a preconditions, post-conditions and completion conditions for each of the KQML CA. Establishing the preconditions, specifies a filter or constrains for triggering event handling. The post-conditions describe the states of the interacting parties assuming successful completion and the completion conditions define the state of what actually happened.

Commitments based upon social conventions. Jones (2003) further discusses that agents' commitments amounts, ultimately, to confusing two kinds of norms called "preservative" and "constitutive". The first are the kind that control antecedently existing activities, e.g. traffic regulation, while the second are the kind that create or constitute the activity itself, e.g. the rules of the game. Hence Jones argues for a model of communication acts based not on intentions, or commitments, but on public conventions.

Semantics for a wider environment. In many MAS models such as the FIPA ACL model, agents are viewed to act in an agent centric model where only other agents reside. In practice, agents must operate within a non-agent computation and networked infrastructure. Agents must also interact with active and passive analogue entities such as humans, buildings and other environment objects. Traditionally, the way this interaction has been designed is that agents must invoke non-ACL interfaces, such as service interfaces to do this, but the semantics of how a sender's mental attitudes perceive operations resulting in inexplicable failures is not defined.

Semantics coverage. Some language constructs such as the Temporal constructs Before and After are implicitly referenced but not formally defined. In addition, the semantics are underspecified in the sense that whilst receiving agents receive CA concerning the intentions and beliefs of the sender, they are free to carry out do internal actions, such as changing beliefs, that can be consistent or inconsistent with the act. Also, the sender agent receives no information that the intended effect of (i.e. goal of, or

reason for doing) that action has resulted (Pitt 1999). Hence Wooldridge (2000) argues that the semantics are not verifiable.

4.2.3 The choice of CAs for the standardised set

FIPA has standardised a core set of CA, the FIPA CA library that it considers useful. In comparison to much of distributed computing that primarily uses a request reply pull type of interaction, it is apparent that FIPA's richer set of communication primitives potentially supports far more flexible interaction. In addition, the semantics of each CA has been specified to act as a mentalistic CA. Examples of types of CA missing from the FIPA core set include:

The lack of expressive CAs to support communication beyond sender mentalistic functions

FIPA CAs are mostly assertives and directives such as queries and exercitives; commisives can be simulated (for example, an agree with the request interaction is used as something akin to promise to try to fulfil the request in the future; there are no permissive or prohibitive type directives that are often used to manage (authorise access to) infrastructure components; there are no declaratives or poetic expressive CA (Singh, 1998).

Labrou et al (1999) argue that FIPA CA to support facilitation such as broker, recommend and recruit are missing. However, since that time (1999) the FIPA CAs propagate and proxy have been included and the brokering and recruitment IPs have been defined.

Although, FIPA CAs include some phatic functions to help manage and control the communication such as refuse, failure, not-understood, agree and cancel several researchers argue that this set should be richer. FIPA for example does not support specifications for basic commands such as poll and check and for controlling modalities such as blocking, immediate, etc. (Charlton, 2000), (Elio, 2005).

KQML has two sub-types of assertive type of CA, tell (a message to create, delete or modify information and reply (a synchronous message to answer an earlier message) whereas the FIPA-CA only includes inform.

KQML has types of CA for managing knowledge such as insert, uninsert, tell and untell, delete one, and delete all.

This is not meant to be an exhaustive list.

4.2.4 CA Set extensibility

CA model of extensibility is promoted by defining new sub-actions rather than defining new CAs, by composing new composite CAs out of existing ones.

4.2.5 CA Use to Share Semantic content

Knowledge engineering applications often focus on the specification of a static shared agreed conceptualisation for a domain, a domain Ontology. It is then assumed that the domain conceptualisation will act as a silver bullet to support interoperability within that domain. In practice, knowledge sharing is messy because there are multiple communities that define over-lapping conceptualisations within a domain, between domains, that conceptualisations become oriented to applications and user viewpoints. Domain models are also living and need to evolve to meet new requirements for their use. For these reasons, defining a domain model is not enough, protocols are needed to support knowledge exchange and its management - the FIPA ACL model is a powerful communication model for knowledge exchange.

4.3 Patterns of CA: IP Model features

4.3.1 Semantics of IP

In practice, an individual CA is used in patterns. A designer of agent systems has to decide whether or not to let the semantics of the individual messages determine the semantics of the conversations versus letting the semantics of the individual messages being determined to some extend by the characteristics of the interaction and able to vary between different conversations. FIPA has decided on the former approach.

4.3.2 IP Flexibility and Extensibility

The simplest way to design interactions is to use pre-specified protocols or interaction stereotypes. Agents can nevertheless engage in meaningful conversation with other agents, simply by carefully following the known protocol that relates to each other's roles in the interaction and organisations. FIPA has standardised a set of stereotypical conversations or IPs that have some limited flexibility so that conversations can be cancelled by the initiator, can be refused or can be failed by other participants. At the application level, the IP model is also extensible because interactions can be nested inside other interactions, for example, one interaction may need to initiate an authorisation with another authority in order to undertake some requested action.

A more flexible approach is to generate interactions on the fly depending on the current status and communications context but this is computationally intensive and often avoided in practice.

4.3.3 Choice of IPs for the standard set

There are several other useful candidates for Standard IPs, for example to support unmediated agent introductions, voting.

In the proposal (that did not complete the path to become a standard) for a FIPA Borda Count IP (Hopkins, 2002), the initiator agent attempts to find a consensus choice that represents the true preferences in a group's election. The participant agents in the group election constitute collective rational behaviour in the sense that they have rankings, which are complete and transitive. The term "Borda Count" derives from the mechanism proposed by Borda [Borda, 1781], who recommended this election system that gave a better representation of what the people really want (better than the 'one man, one vote' system and the pairwise comparison). Using the Borda Count mechanism means that, in principle, points are allocated to a collection of alternative strategies. In a collection of X alternatives, X points will be allocated to the most preferred strategy, X-1 to the next best, and so on down to the least preferred strategy, which is allocated one point. The protocol requires that all voters have to rank their preferences among the X alternatives, except if the Borda Count calculator decides otherwise. The protocol is used then at a central location to add up the allocated points. The preferences are collected centrally to rank the scores given to each strategy, and to select the strategy with the maximum score as the winner. This specification presents a version of the Borda Count mechanism in which co-operating agents find a most preferred choice within a set of alternatives.

Many distributed computing models including MAS models require the used of mediators such as directory services and brokers to match service requestors to service providers and mediate between the two during the service. Whilst this level of an indirection has the advantage that it support dynamic service provider user matching and can provide a well-know contact point for services, these same two characteristics can also be viewed as disadvantages in these sense that the contact point can be designed to be a single point of failure and it adds another level of indirection of asking a mediator for information about another agent characteristics. If an acquaintance or hello interaction protocol, an agent could be introduced to or ask another about its characteristics without going through a third-party. There are many situations where this would be beneficial because two agents may want more

privacy when interacting or one agent may be interested in cross-checking about each others' characteristics since they last met.

Variations of particular protocols such as CN have been proposed (Give refs here). Need to decide how to deal with variations of IPs.

Phatic communication aspects such as the effects of cancelling actions, asynchrony, abnormal or unexpected IP termination and how to explicitly signal the switch to nested IPs are not explicitly addressed. There is also a issue concerning the semantics of the cancel CA this is defined to cancel any single CA that has duration. This causes problems when we want the CA to affect a IP that has duration that is made up of individual CAs that have no significant duration, we can't cancel (prevent from completing) an act that is already done, e.g., register instance of the request CA in the Subscription IP. IPs may need to be designed at an individual level to be reversible.

4.3.4 IP Model Notation and Expressivity

Although the AUML has a level of expressivity for modelling agent interaction, others have argued such as (Cranfield, Purvis ???) that the guard conditions for key stages of the interaction are not adequately defined and that other models such as Coloured Petri-nets may be a better option for this. However, the inherent parallelism of Petri-nets makes them difficult to verify (Chaib-Draa and Dignum, 2002) A further concern is that the AUML is not in a declarative form for computation and for the dynamic specification and distribution of new interaction patterns.

4.4 Architectural and Service Model features

The service model should support open services. This extensibility and openness is desirable at two distinct levels of granularity: at the agent level and at the agent component level, the service components that underpin the agent. An important design decision is whether to model a service as groups of agents, as single agents or as agent components. 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 FIPA specifications, the message transport is a non-agent service.

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, 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 transports specification and the Abstract Architecture specification. can support multiple message transport protocols.

There are several restrictions of the current agent management and middle services. Many MAS models such as the Abstract Architecture model mandates the use of a directory service but they not be needed for peer-to-peer action. The agent management interface is defined using a frame-based Ontology. The current directory service modelled as the DF agent in the agent management specification does not support additional types of query or subscribe interaction, DFs can't be federated, and support service agent registrations of other types of agent o service agents such as user agents. The

Service ontology for an agent specified the interaction protocol an agent supports but it does not include the agent role within the interaction. The DF service ontology also does not define competence or reliability of the service provider

Transport issues include the assumption of a reliable transport so that messages cannot get out of order. Fields such as an in-reply-to field has been added to the ACL message header to support message management by defining an identifier for an instance of IPs, there is no standard syntax for structure for such fields specified.

5 Uncompleted FIPA specifications and Candidates for future FIPA Standard specifications

Developing specifications for standardisation is often a somewhat risky process in practice. Many specifications have been proposed in many different standards organisations that become obsolete and do not gain a critical mass of users. Lack of adherence to complete the process of turning a specification into a standard is often a reason why many worthy ideas for standardisation of specifications – sometimes there is a lack of commitment to see proposed ideas to completion to a standard. Other reasons include the time may not be right or that design options may be controversial and the specifiers may not come to a agreement. Specifiers may be more interested in specifying specifications rather than also developing reference implementations for them.

Todo : A classification of such models: non semantic interfaces and architectures for non-agent software

5.1 Semantics

5.1.1 Semantics based upon linguistic approach

5.1.2 Semantics based upon an institutions and policies

5.2 Agent management

5.2.1 Agent Security management

5.2.2 Agent Configuration Management

5.3 Mobile Agents (MA)

The focus in this article is what were the past experiences by FIPA in the realm of mobile agent, how does the concept of a mobile agents interlink with other FIPA specifications and what is the motivation for a new effort to develop mobile agents. In the past, mobile agents has been regarded by some researchers as a disjunctive type of agent to the intelligent communicating, FIPA, agent type that is static [TODO, ADD REF]. However mobile agents need to communicate with each other and the ACL model is a useful model to support this and ACL agents could advantageously utilise agent mobility, e.g.,, some aspects of agent mobility such as agent invocation could be abstracted to more general agent management models for agent configuration. In [FIPA0087], agent mobility was supported using an a management service specific ontology, the Agent Mobility Ontology, rather than being defined at

the CA level in terms of new mobility specific CAs. Not also that FIPA has separated support for agent migration, cloning and invocation in [FIPA0087] from support for ubiquitous computing type agents and mobile devices such as in the FIPA Nomadic Application Support Ontology Specification [FIPA0014] .

5.3.1 Basic Concepts

Mobile Agents are agents that are able to migrate from one host to another. Mobile Agents are able to communicate with each other. Agent Platform provides access to local resources through services. Mobile agents are often modelled at the implementation level to consist of code, data, state and meta-Information. There are different types of agent mobility: agent migration, mobile agent states; mobile devices and mobile computation. Agent migration

Mobile agent states is often used with agent cloning of the agent clone, a cloned agent can then be imprinted with an agent's memory or states. Mobile devices are more portable, low resource computation platforms that enable users to access remote services on route using a wireless connection. Mobile computation is similar to remote access and allows one system to run a computation on another system to utilize resources on remote system.

The motivation for Mobile Agents is that they can provide significant advantages regarding: using disconnected operations, use of temporary short on-line network connections leading to lower bandwidth consumption, low latency interaction; complex computations can be offloaded or delegated and it is natural kind of software distribution.

The open issues and future work for mobile agents includes: Migration between heterogeneous platforms; Interoperability for mobile code; Security concerns of mobile code; Representation of mobile code; How to develop mobile agents.

5.3.2 FIPA's past experiences with Mobile Agents

FIPA activities for Mobile Agents started in 1998 and were considered to be a sub-part of agent management (FIPA TC1). A first specification was produced at the end of 1998 called the "FIPA Agent Management Support for Mobility Specification" This was updated as part of the 2002 drive towards FIPA standards [FIPA0087] but this did not become a standard because of a lack of a reference implementation. [FIPA0087] is similar to the 2000 OMG Mobile Agent Facility (MAF) specification but is a more abstract model that can be grounded using MAF. The main features of [FIPA0087] are:

- Optional: Mobility features do not need to be supported by an agent or agent platform.
- Abstraction: This specification does not mandate: ·the use of any explicit technology for supporting mobility.or any implementation; Mobile agent security is not currently addressed by this specification. ·The specification defines extensions that are necessary to the AMS to support mobility.
- Simple Mobility vs. Full Mobility Protocols: An agent can use a high level protocol and delegates mobility to an agent platform AP vs. An agent directs the mobility protocol itself and does not delegate responsibility to the AP.
- Mobility Life Cycle: [FIPA0087] specification extends the existing life cycle given in [FIPA00023] by adding a new state (Transit) and two new actions to enter and leave that state (Move and Execute).
- Mobility Protocols: a number of standard protocols were defined to cover various forms of agent mobility such as: Agent migration (agent transport between two agent platforms),

Agent cloning (self-copy onto a given platform), and Agent invocation (creation of an agent on a given platform).

- Agent Profiles: Since a mobile agent can be transported between APs in a variety of formats it can make a number of demands upon an AP for a required set of conditions to be met before such an agent can be executed. Each of these dependencies can be expressed as part of the meta-information of a mobile agent within the :profile parameter.
- Agent Mobility Ontology: this uses a frame-based representation and extends the FIPA-Agent-Management ontology defined in [FIPA00023]. It defines the agent mobility protocols actions such as move, transfer; protocol exceptions such as mobility-unsupported, profile-unsupported, agent-already-present; Mobile agent profiles

5.3.3 Features and Limitations of the old FIPA MA model and suggested improvements

Current FIPA model only partially covers the main challenges of mobile agent systems development. Regarding code and data relocation [FIPA0087] provides two mobility protocols (simple mobility and full mobility), and three mobility cases (agent migration, agent cloning, and agent invocation). All these protocols assume that the moving agent initiates the mobility process. There are certain cases, however, where the process may need to be initiated by the agent platform where the moving agent resides. For instance, in nomadic applications, it is not unusual to have some agents running on a resource-limited mobile host (i.e. a laptop, PDA...). If this host is running out of battery or is about to lose its connectivity, agents running on it could be transferred to an agent platform on an alternative host to allow them to continue execution. Though this could be achieved via direct request to the involved agents, a transfer mobility case where an agent platform may accomplish the migration in a transparent way seems to be more appropriate. This could involve extensions to both [FIPA00014] and [FIPA00087], and some contributions to the P2P Nomadic Agents Working Group.

Issues like agent location and agent tracking are not covered by [FIPA0087]. How to discover the platform where an agent resides, or how to make a message to reach an agent who is migrating or has already migrated to another platform when the message arrived, or simply how to locate an agent by its AID, or simply how to uniquely name a mobile agent are issues that need to be addressed. Tracking of mobile agent itineraries may be needed for certain applications.

At the moment, cloning is considered just a special case of mobility, where the agent at the source platform does not terminate when its code and state are transferred to the destination agent platform. However, cloning may be used to provide state integrity and consistency between hops of a mobile agent. An agent may migrate from its home agent platform (HAP) to another leaving a clone at the source platform. The clone at the HAP may remain suspended as long as the migrating agent is “alive”, and resume execution if it is destroyed. After the migrating agent has completed the task that originated the migration, or between the different hops within the agent route, both instances of the agent may be synchronized to address any status inconsistencies.

[FIPA0087] mobility protocols assume that, when an agent moves, “agent code (and possibly state)” is transferred from the source agent platform to the destination. Actually, there may be cases where code transfer is not necessary, if the moving agent code is available at the destination platform or that platform has means of getting it from another source. [FIPA0087] mobile agent description should be extended to allow including not only the code-base of the agent, but also how to access it from available repositories. This is important in nomadic applications, where agents may need to be transferred from mobile devices, which are usually battery and bandwidth limited.

Agent profiles defined in [FIPA0087] allow expressing requirements over an agent platform to execute a given mobile agent. Current profile specification allows expressing requirements based on the agent system (type and version), the language the mobile agent is coded in, and the operating system of the destination agent platform. In some cases, however, an agent may need to impose higher-level requirements over the platform it is moving to. In particular, one of the key benefits of mobile agents is the possibility of taking advantage of locality of services to make an efficient usage of resources. For instance, an agent needing to retrieve a large amount of data from a database may move to the host where the database resides and perform its queries locally. Considering that, on agent-based systems, access to local resources on machines will most likely be performed through services provided by stationary agents on those machines, mobile agent profile specification could be extended to allow to express (maybe through the use of a :service parameter) which services the agent need to be provided at the platform.

Finally, [FIPA0087] specification does not cover security at all. The use of mobile agents raises numerous security considerations [Jansen and Karygiannis, 2000], and these considerations need to be addressed before any solution based on mobile agents can be deployed in real, production environments. Traditionally, security in multi-agent systems has been regarded as a secondary issue to be developed once the system was complete, or has been assumed to be provided by lower or upper layers in the architecture. However, security in mobile agents is complex and critical enough to need to be considered at the first stages of the specification. Furthermore, security protocols and policies must be standardized to allow different platforms to interoperate.

5.4 Ubiquitous Computing

5.4.1 Basic concepts

The focus of distributed computing is shifting away from manual one-to-one client-server type human personal computer interaction to include more varied interaction, such as peer to peer, ad hoc and mesh type interaction, with a wider variety of heterogeneous, fixed, mobile and untehered computers and communication devices, that are possibly embedded, that range in size and ability. This latter type of ubiquitous computing interaction highlights added challenges for MAS design.

5.4.2 FIPA's past experiences with UC

Confronted with these circumstances of accessing heterogeneous high and low resources devices and networks, the nomadic end-user would benefit from having the following functionality provided by the infrastructure: information about expected performance, agent monitoring and controlling the transfer operations, and adaptability. The ability to automatically adjust to changes in a transparent and integrated fashion is essential for nomadicity; nomadic end-users are usually professionals in areas other than computing. Furthermore, today’s mobile computer systems are already very complex to use as productivity tools. Thus, nomadic end-users need all the support that a FIPA agent-based distributed system can deliver and adaptability to the changes in the environment of nomadic end-users is an important issue.

During 2000-20002, several MAS research projects were initiated that investigated the use of agents to enhance mobile, wireless applications and services. Two prominent ones were the EU FP5 projects, CRUMPET and LEAP that lead to the development of MAS designs and implementations such as Micro-FIPA-OS and JADE-LEAP [TODO ADD REFS] for use in mobile and wireless environments. These projects also led to the development of several standard specifications:

- [FIPA0069] FIPA ACL Message Representation in Bit-Efficient Specification
- [FIPA0088] FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification

- [FIPA0014] FIPA Nomadic Application Support Specification
- FIPA0091] FIPA Device Ontology Specification.

One main focus was to develop encodings for ACL message that could be used in lower bandwidth wireless environments, the so called bit-efficient encodings. Whereas the Nomadic Application support specification defines agent middleware to monitor and control a Message Transport Protocol (MTP) and the underlying Message Transport Connection (MTC) so that content can adapt to the QoS of the available network. The FIPA device ontology (specification) can be used by agents when communicating about devices to provide the information to enable adaptation to device characteristics to take place.

In addition, the following specifications were produced that did not make standard status:

- [FIPA00095] FIPA Agent Discovery Service Specification
- [FIPA00096] FIPA JXTA Discovery Middleware Specification
- [FIPA00092] FIPA Message Buffering Service Specification

5.4.3 Features and Limitations of the old FIPA UC model and suggested improvements

Todo.

5.5 Human Agent Interaction (HAI)

5.5.1 Basic concepts

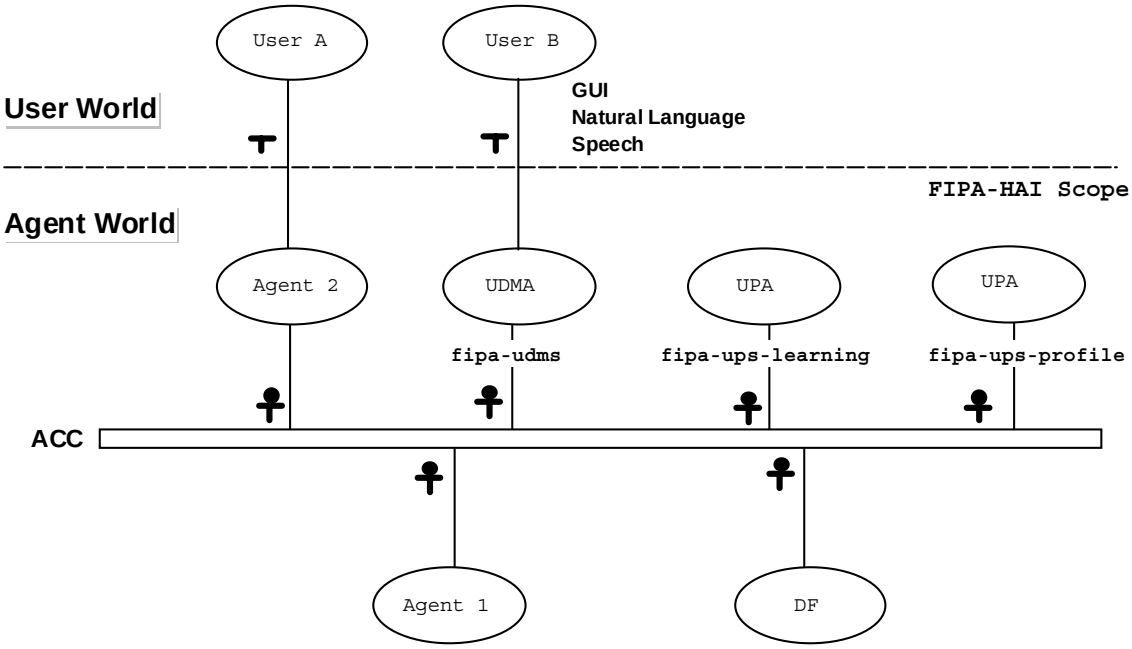
There are many possible focuses for HAI that can make the field quite a broad one:

1. Human-agent communication for decision making
2. Learning
3. Interfaces to gather human input and convert them into some agent language
4. Personal Agent: an assistant that acts on behalf of a human owner
5. *User Agent*: An agent which interacts with a human user.
6. *User Model*: A user model contains assumptions about user preferences, capabilities, skills, knowledge, etc, which may be acquired by inductive processing based on observations about the user. User models normally contain knowledge bases which are directly manipulated and administered.
7. *User Dialog Management Service* (UDMS): An agent service in order for FIPA agents to interact with human users; by converting ACL into media/formats which human users can understand and vice versa, managing the communication channel between agents and users, and identifying users interacting with agents.
8. User *Personalization* Service: An agent service that offers abilities to support personalization, e.g. by maintaining user profiles or forming complex user models by learning from observations of user behavior.

5.5.2 FIPA's past experiences with HAI

FIPA activities with HAI started in 1998 in FIPA TC8, it focuses on points 7 and 8.. A first specification was produced at the end of 1998 called the " Human-Agent Interaction " This was updated as part of the 2002 drive towards FIPA standards [FIPA0004] but this did not become a standard because of a lack of a reference implementation.

Figure 15 illustrates the different entities and relationships between entities that are considered crucial to the human-agent interaction process. The figure mainly consists of two parts: on the upper side (which may be regarded as the "user world") there is the user and the interfaces that are available to the user, while on the lower side, we have the "agent world" where agents operate and communicate [FIPA0004].



ACL Interface

non-ACL Interface

fipa-udms	FIPA user dialogue management service ontology
fipa-ups-profile	FIPA user personalization profile ontology
fipa-ups-learning	FIPA user personalization learning ontology

Figure 15: Human-agent interaction reference model [FIPA0004]

In the user world, a User Dialog Management Service (UDMS) provides two interfaces. One is user interface for human user which serves as interface to a device: a graphical user interface to a computer

with its direct manipulation possibilities, a voice interface to a mobile phone, a gesture-based interface to a PDA, etc. Another is agent interface which interacts with agents using the ACL. So, user dialogue management service translates between user action and the ACL. Internal process of the system is due to implementation, FIPA does not standardize a specification of it. Thus, from the point of view of agents, interacting with the users is not different from interacting with agents [FIPA0004].

In the Agent World, interaction is possible between users and agents via Via UDMS. Note that no specific functionality is associated with the term "user agent". Any agent that interacts with a human user shall be subsumed by this term. A crucial part in the intelligent and user-supportive behavior of an agent is played by the model of its user. A user model contains assumptions about user preferences, capabilities, skills, knowledge, etc, which may be acquired by inductive processing based on observations about the user. User models normally contain knowledge bases which are directly manipulated and administered. In the model, as illustrated in Figure 15, a User Dialog Management Agent (UDMA) interacts with a User Personalization Agent (UPA) via ACC in order to delegate the tasks of user model acquisition, representation, and provision. This agent offers its services to the whole agent world. In particular, one of such services concerns user model learning, which may be exploited to form knowledge about the user from observations, while the other concerns user profiling, i.e., maintaining explicitly formulated knowledge about the user in data-base- or knowledge-base-like formats [FIPA0004].

Each of the main components of the HAI model, the UDMA and UPA have use-cases defined, e.g., One UDMA can interact with one user, multiple UDMA can interact directly with one user, perhaps because the user wants to use multiple types of UI, or a user can use a broker to interact with multiple UDMAs, finally the multiple UDMAs can interact with multiple users.

5.5.3 Features and Limitations of the old FIPA HAI model and suggested improvements

The HAI agents such as the UDMA are defined in terms of a frame-based ontology, `fipa-udms` and use existing FIPA CAs rather than defining new CAs. Several key challenges for the design of HAI are:

- How much to represent and constraint human natural language so that it can be mapped to agent interaction: FIPA0004 uses a SKDL, Structure Knowledge Description Language but it is not clear that the process is for converting natural written and spoken text into SKDL; the mapping from SKDL to FIPA-ACL/content language and Ontologies are not clear; multi-lingual aspects and internationalisation are not considered.
- How to map between user dialogues and agent dialogues: an example is given in FIPA0004 but no explicit process is given .
- How to interleave human interaction with agent interaction.

5.6 Services and SOA

6 FIPA and its relationships with other distributed computing standards

6.1 XML Web and Web-services

6.2 GRID

6.3 Semantic Web and Web Services

6.4 IETF TCP/IP

7 End Discussion

7.1 Summary of the Features and Strengths of the FIPA MAS model

We end this section by summarising the strengths and features of the FIPA model. The features of the FIPA ACL model are:

1. Set of MAS design models that can lead to computation models of logic-based and semantic models, that are abstract and flexible enough to be independent of specific technologies but yet able to be grounded using specific technologies
2. Specifications of a rich set of CA, communication primitives that support information sharing information created, information queries and task sharing.
3. There is a Formal semantics to define each CA and some computational models of these have been built and tested, although most computation models of the CAs rely on the semantics of the CAs to relate to the pattern of use of the CA
4. Specifications of Interaction Patterns of the CAs that support cooperative and competitive, push and pull interaction, one to one and one to many interaction, information and task sharing.
5. Specifications of a generic Architecture model and service model.
6. Specifications have been tested in practice and demonstrated that they enable interoperability and open service invocation.
7. Holistic framework that interlinks semantic knowledge-based content with a semantic communication protocols and communications context for exchanging the content
8. Development life-cycle for specifying, experimenting with implementations and standardising mature implementations,
9. A range of tools including open source ones that implement the specifications
10. Widely deployed specifications have been used in numerous applications and projects. See appendix A.

7.2 Future work

There are several opportunities to improve features of the FIPA specifications that for example restrict the ability to: synthesise new CAs and IPs, to be able to statically and dynamically vary the semantics of the CAs and IPs and to specify more flexible agent middleware service interaction in directory services. Further theoretical work is needed that is tied to further experimental validation if these issues are to addressed in practice.

For a MAS model that specifies the interoperability of agents to be become widely deployed, the trade-off between theoretical and pragmatic issues must be carefully balanced. It needs to consider the concerns requirements of and to support multiple stake-holders, not just the theoreticians that develop the underlying theoretical models but also the computation model specifiers, application and tool developers and business and academic end-users. There needs to life-cycle to allows theoretical and pragmatic models to propose specifications, that can be evaluated in practice as well as theoretically, that can identify those specifications with effective computation models and mature through iterative interaction.

Methodologies

Pervasive computing agents: need greater autonomy, need to have mutual models of the situated environment.

(Web) Service Agents

Human

Others.

7.3 Recommendations

The following recommendations are made:

That FIPA develop a specification of CAs that can support multiple heterogeneous types of agent communication to support the exchange of knowledge, mental attitudes, human interaction, non-agent computation and network interaction. This implies that MAS need a multi-lateral view of CA semantics rather than a single one such as mentalistic attitudes.

That FIPA should specify a process of developing domain specific MAS models in practice as a complementary life-cycle development to a top-down process of working from a general abstract MAS architecture, to help identify the theoretical MAS models that are usable and useful in practice.

That FIPA should develop specifications to aid the design, implementation, reconfiguration, maintenance and management of MASs.

Enable parameters and constraints of communication protocols to be explicitly modelled such that more flexible and richer interaction can occur.

7.4 Final conclusion

Standard communication specifications naturally have heir critics. Often, there is a variety of stake-holder interests in specifying standards leading to standards that may be either considered to be too expressive or not expressive enough for designers and implementers to use or that are difficult to embed in existing infrastructures. In addition, standards may need adjustment or not work well in specific applications. There are also those who argue that standards may not be able to always guarantee consistent design and interoperability. These points are true for standardisation in general, not just for MAS standards.

However, these challenges should not distract from the benefits of standards as a key enabler to support interoperability and open service interaction in practice and to lead to a critical mass of users and uptake. Good standardisation is about striking an optimal balance between developing expressive, flexile, abstract models of key behaviours versus being able to reify models in a constrained way, to successfully deploy them.

8 References

Agentcities.RTD Project Deliverable D1.4 (2002) Final Report. Available from http://www.agentcities.org/EURTD/, accessed 2006-04-01.

Bellifemine, F., Poggi, A., Rimassa, G. (1999) JADE - A FIPA-compliant agent framework. Proc. PAAM'99, London, April 1999, 97-108.

Bradshaw, J.M., et al. (2004) Making Agents Acceptable to People. In Intelligent Technologies for Information Analysis: Advances in Agents, Data Mining, and Statistical Learning, N. Zhong and J. Liu, eds., Springer-Verlag, , pp. 355–400.

Charlton, P, Cattoni, R, Potrich, A and Mamdani, E, (2000) Evaluating the FIPA Standards and its Role in Achieving Cooperation in Multi-Agent Systems. In: Multi-Agent Systems, Internet and Applications, HICS-33 Software Technology Minitrack, USA.

Cranefield, S., Purvis, M., Nowostawski, M., and Hwang, P. (2005) Ontologies for Interaction Protocols", In *Ontologies for agents: theory and experiences*, V. Tamma, S. Cranefield, T. Finin, and S. Willmott (eds.). Basel, Birkhäuser (Whitestein Series in Software Agent Technologies), 1–17

Elio, R. and Petrinjak A. (2005) Normative Communication Models for Agent. Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, 11(3): 273-305.

Ehrler, L. and Cranefield, S. (2004) Executing Agent UML diagrams, Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04), ACM Press, 906–913.

[FIPA0001] Standard Specification No. SC00001, FIPA Abstract Architecture Specification. Available from http://www.fipa.org/specs/fipa00001/index.html

[FIPA0087] Obsolete Specification No. O00004, FIPA Human Agent Interaction Specification. Available from http://www.fipa.org/specs/fipa00004/index.html

[FIPA0008] Standard Specification No. SC00008, FIPA SL Content Language Specification. Available from http://www.fipa.org/specs/fipa00008/index.html

[FIPA0014] Standard Specification No. SI00014, FIPA Nomadic Application Support Specification. Available from http://www.fipa.org.

[FIPA0023] Standard Specification No. SC00023, FIPA Agent Management Specification. Available from http://www.fipa.org.

[FIPA0026] Standard Specification No. SC00026, FIPA Request Interaction Protocol Specification. Available from http://www.fipa.org.

[FIPA0027] Standard Specification No. SC00027, FIPA Query Interaction Protocol Specification. Available from http://www.fipa.org.

[FIPA0028] Standard Specification No. SC00028 , FIPA Request When Interaction Protocol Specification

[FIPA0029] Standard Specification No. SC00029, FIPA Contract Net Interaction Protocol Specification. Available from http://www.fipa.org.

[FIPA0030] Standard Specification No. SC00030, FIPA Iterated Contract Net Interaction Protocol Specification. . Available from http://www.fipa.org.

[FIPA0033] Standard Specification No. SC00033, FIPA Brokering Interaction Protocol Specification. Available from http://www.fipa.org.

[FIPA0001] Standard Specification No. SC00034, FIPA Recruiting Interaction Protocol Specification. Available from http://www.fipa.org.

[FIPA0035] Standard Specification No. SC00035, FIPA Subscribe Interaction Protocol Specification. Available from http://www.fipa.org.

[FIPA0036] Standard Specification No., SC00036, FIPA Propose Interaction Protocol Specification. Available from http://www.fipa.org.

[FIPA0037] Standard Specification No. , SC00037, FIPA Communicative Act Library Specification. Available from http://www.fipa.org.

[FIPA0061] Standard Specification No., SC00061, FIPA ACL Message Structure Specification. Available from http://www.fipa.org.

[FIPA0067] Standard Specification No., SC00067, FIPA Agent Message Transport Service Specification. Available from http://www.fipa.org.

[FIPA0069] Standard Specification No., SC00069, FIPA ACL Message Representation in Bit-Efficient Specification

[FIPA0070] Standard Specification No., SC00070, FIPA ACL Message Representation in String Specification. Available from http://www.fipa.org.

[FIPA0071] Specification No., SC00071, FIPA ACL Message Representation in XML Specification. Available from http://www.fipa.org.

[FIPA0075] Standard Specification No. SC00075, FIPA Agent Message Transport Protocol for IIOP Specification. Available from http://www.fipa.org.

[FIPA0084] Standard Specification No., SC00084, FIPA Agent Message Transport Protocol for HTTP Specification. Available from http://www.fipa.org.

[FIPA0085] Standard Specification No. SC00085, FIPA Agent Message Transport Envelope Representation in XML Specification. Available from http://www.fipa.org.

[FIPA0087] Deprecated Specification No. DC00087, FIPA Agent Management Support for Mobility Specification. Available from http://www.fipa.org/repository/fipa98.html

[FIPA0088] Standard Specification No. SC00088, FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification. Available from http://www.fipa.org.

[FIPA0009] Standard Specification No. SI0009, FIPA Device Ontology Specification. Available from http://www.fipa.org.

[FIPA0094] Standard Specification No. SC00094, FIPA Quality of Service Specification. Available from http://www.fipa.org.

Hopmans, G. and Braspenning, P. J. Reaching consensus in agent systems with the Borda Count Interaction protocol. A proposal for a new FIPA Interaction Protocol (2002) Multi-agent Interoperability MAI'02 Workshop at the 25th German Conference on Artificial Intelligence (KI2002), Aachen, Germany.

Jansen, W., Karygiannis, T.: Mobile agent security. NIST Special Publication 800-19, National Institute of Standards and Technology (2000)

Núñez-Suárez J., O'Sullivan, D., Brouchoud, H., Cros P., Moore C , Byrne Ciara (2000). Experiences in the use of FIPA agent technologies for the development of a personal travel application. Proc. 4th Int. Conf on Autonomous agents, 357 - 364 .

Jones, A. J. I. and Parent X. (2003) Conventional Signalling Acts and Conversation. Workshop on Agent Communication Languages. In (Dignum F., Eds) Advances in Agent Communication: International Workshop on Agent Communication Languages, LNCS Vol. 2922, 1-17.

Labrou, Y. Finin, T. and Peng. Y. (1999) The current landscape of agent communication languages. Intelligent Systems, 14(2):45-52.

Louis, V. and Martinez, T. (2004) An operational model for the FIPA-ACL semantics". Proc. AAMAS'05 Workshop on Agent Communication (AC2005), Utrech, 2005.

Nwana, H., Ndumu, D., Lee, L., Collis, J.. (1999) ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems. In Applied Artifical Intelligence Journal, Vol. 13, No. 1, 129-186.

Odell, J., Van Dyke, P.H. and Bauer, B. (2001) Representing Agent Interaction Protocols in UML. In: Agent-Oriented Software Engineering, Ciancarini, P. and Wooldridge, M., Eds., Springer, pp. 121-140, Berlin

Pitt, J. and Mamdani, A. (1999a) Some Remarks on the Semantics of FIPA's Agent Communication Language. Autonomous Agents and Multi-Agent Systems, Kluwer Academic, 2(4): 333-356.

Pitt J.V. and Bellifemine. (1999b) F. A Protocol-Based Semantics for FIPA'97 ACL and its Implementation in JADE. Proceedings 6th Congress of the Italian Association for Artificial Intelligence AI*IA'99.

Poslad S. History of FIPA (2005) Available online via http://www.fipa.org/subgroups/ROFS-SG.html, accessed on 2006-12.

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

Reed, C. A., Norman, T. J. and Jennings, N. R. (2002). Negotiating the Semantics of Agent Communication Languages. Computational Intelligence 18(2), 229-252.

Sadek M.D. (1992) A study in the logic of intention. Proc. 3rd Conf on Principles of knowledge representation and reasoning, KR'92, Cambridge MA, 462-473.

Singh, M.P. (1998) Agent communication Languages: rethinking the principles. IEEE Computer, 31(12): 40-47.

Singh, M.P. and Huhns, M.N. (2005) Service-Oriented Computing: Semantics, Processes, Agents. John Wiley & Sons, Ltd.

Willmott, S. Calisti, M., Rollon E. (2002) Challenges in Large Scale Open Agent Mediated Economies, Proceedings of Workshop on Agent Mediated Electronic Commerce, AAMAS 2002.

Wooldridge, M. (2000) Semantic Issues in the Verification of Agent Communication Languages. Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, 3(1): 9-31.