AGENT- COMMUNICATION LANGUAGES:

FIPA

Mario Gómez
IIIA-CSIC
mario@iiia.csic.es

1 INTRODUCTION

1.1 Agent architectures

- Individual agents
- Mediated architectures: based on client-server architecture
- Markets and swarms: *emergent behaviour*
- MAS (Multi Agent Systems), with an emphasis on
  - agent-to-agent communication
  - cooperation and collaboration
  - team and coalition formation
  - information sharing among the team
  - joint beliefs, goals and plans

1.2 Social ability, communication and cooperation:

- ***Social ability*** in agents is the ability to interact with other agents (and possibly humans) via some kind of *agent-communication language*, and perhaps cooperates with others.
- **Communication** is important because:
  - Communication is a requirement for cooperation.
  - Societies can do things that no individuals can. Some goals need cooperation.
  - Diversity introduces heterogeneity.
  - Autonomy encourages disregard for other agents' internal structure
- Communicating agents need to understand a "common language":
  - understanding of its vocabulary
  - knowing how to effectively use the vocabulary to perform task and achieve goals.

1.3 The intentional stance

- Dennet's ladder of personhood, six conditions:
  - *rationality*
  - *intentionality*
  - *stance*
  - reciprocity
  - communication
  - consciousness

- The first three are mutually interdependent and are used for defining a basic *intentional system*
  - *Intentionality* refers to the system's intentions as rationally following from its beliefs about the world and from its high level preferences
  - The ***intentional stance*** refers to modelling a system as an intentional system, i.e. as having beliefs, intentions and so forth. Prediction from intentional stance assumes rationality
- The intentional stance is an abstraction tool, which provide us with a convenient and familiar way of describing, explaining and predicting the behaviour of agents, without having to understand how the mechanism actually works.
- It gives us the potential to specify systems that include representations of other systems (*reciprocity*). Such nested representations are essential for agents that must cooperate with other agents.
- Reasoning agents (agents as KBS, symbolic AI paradigm)
  - Contains an explicitly represented, symbolic model of the world
  - Makes decisions(i.e. about what actions to perform) via symbolic reasoning

1.4 Agent Oriented Programming (AOP)

- Much of the interest in agents from AI community has arisen from Shohan's notion of AOP [Shohan, 1993]
  - New programming paradigm, based on a societal view of computation
  - Key idea is that of directly programming agents in terms of intentional notions like belief, commitment, and intention.
  - Motivation: in the same way we use the intentional instance to describe humans, it might be useful to use the intentional instance to program machines.

1.5 Agent theory: Speech Act Theory and BDI theories

- Speech acts theories are pragmatic theories of language: how utterances are used to achieve goals and intentions [Austin, 1962].
- **Speech acts** may be understood in terms of an intentional-level description of an agent: references to beliefs, desires, intentions (BDI) and other modalities.
- BDI architectures describe the internal structure of an agent by the mental states of *beliefs*, *goals* and *intentions*.
- *Communication* is a means to
  - reveal to others what our BDI state is and
  - attempt to effect the BDI state of others.
- *Recursivity*: an agent has beliefs about the world, about other agents, about the beliefs of other agents, about the beliefs another agent has about it, …

- Speakers do not just utter true or false sentences, they perform speech acts: requests, suggestions, promises, threats, etc.
- Different types of speech act [Searle, 1969]:
  - *representatives*: such as informing, e.g. 'it's raining'
  - *directives*: attempts to get the hearer to do something, e.g. 'please make the tea'
  - *commisives*: which commit the speaker to doing something, e.g. 'I promise to…'
  - *expressives*: whereby a speaker expresses a mental state, e.g. 'thank you!'
  - *declarations*: such as declaring war or christening.
- In general, a speech act can be seen to have two components:
  - a *performative verb*, e.g. request, inform, etc.
  - propositional *content*:, e.g. "the door is closed"
- Examples of different speech acts with the same content but different performatives:
  - content = "the door is closed"
  - performative = request speech act = 'please close the door'
  - performative = inform speech act = 'the door is closed!'
  - performative = inquire speech act = 'is the door closed?'

1.6 Communication levels

- *Transport level*: messages are transported using some lower-level transport protocol (SMTP, TCP/IP, http, etc.).
- *Message level*: an ACL defines the types of messages (and their meaning) that agents may exchange.
- *Conversation level*: interaction protocols (negotiation, auction, etc.), define task-oriented, shared sequences of messages.
- Higher-levels: conceptualisation of an agent's goals and strategies drives the agent's communicative (a non-communicative) behaviour.

1.7 AGENT0 and PLACA

- The first AOP language
  - Implemented as an extension to LISP
- Components of an agent
  - a set of capabilities (things the agent can do)
  - a set of initial beliefs
  - a set of initial commitments (things the agent will do); and
  - a set of *commitment rules*.
- Each commitment rule contains
  - a *message condition*
  - a *mental condition*; and
  - an action.

- On each 'agent cycle'…
  - the message condition is matched against the messages the agent has received
  - the mental condition is matched against the beliefs of the agent
  - if the rule fires, then the action gets added to the agents commitment set
- Actions may be
  - *private*: corresponding to an internally executed subroutine, or
  - *communicative*, i.e., sending messages.
- Messages
  - 'requests' to commit to action
  - 'unrequests' to refrain from actions
  - 'informs', which pass on information
- Example of commitment rule

```
COMMIT (
(agent, REQUEST, DO (time, action)),       ;;; msg condition
(B,
     [ now, Friend agent] AND
     CAN (self, action) AND
     NOT [ time, CMT (self, anyaction)] ),  ;;; mental condition
Self,
DO (time, action))
```

- PLACA [Thomas, 1995] is a more refined implementation intended to address one sever drawback of AGENT0: the inability of agents to plan and communicate requests for action via high-level goals.

1.7.1 Concurrent METATEM

- Is a multi-agent language in which each agent is programmed by giving it a *temporal logic* specification of the behaviour it should exhibit [Fisher, 1994].
- Temporal logic is classical logic augmented by *modal operators* for describing how the truth of propositions changes over time.
- Modal operators: $\Box$ always $\diamond$ sometime in the future $\blacklozenge$ sometime in the past $\bigcirc$ tomorrow(in the next state)
- METATEM is a framework for directly executing *temporal logic* specifications.
- Gabbay's separation theorem: any arbitrary logic formula can be rewritten in a logically equivalent *past* $\Rightarrow$ *future* form.
- This *past* $\Rightarrow$ *future* form can be used as *execution rules*. A METATEM program is a set of such rules
- Execution proceeds by a process of continually matching rules against a "history", and *firing* those rules whose antecedents are satisfied.
- The instantiated future-time consequents become *commitments* that must be satisfied.
- Concurrent METATEM provides an operational framework through which societies of METATEM processes can operate and communicate

- Example: Snow White has some sweets (resources), which she will give to the dwarves (resource consumers).
  - She will always eventually give to a dwarf that asks.
  - She will only give to a dwarf at a time.

Snow-White (ask)[give]:

$$\bullet\!\!\!\circ \text{ ask } (x) \Rightarrow \diamond \text{ give } (x)$$

$$\text{give } (x) \wedge \text{give } (y) \Rightarrow (x = y)$$

eager (give)[ask]:

$$\text{Start} \Rightarrow \text{ask (eager)}$$

$$\bullet\!\!\!\circ \text{ give (eager)} \Rightarrow \text{ask (eager)}$$

greedy (give)[ask]:

$$\text{Start} \Rightarrow \Box \text{ ask (greedy)}$$

courteous (give)[ask]:

$$((\neg \text{ask (courteous) } S \text{ give (eager)}) \wedge$$
$$(\neg \text{ask (courteous) } S \text{ give (greedy)})) \Rightarrow$$
$$\text{ask (courteous)}$$

shy (give)[ask]:

$$\text{start} \Rightarrow \diamond \text{ ask(shy)}$$

$$\bullet\!\!\!\circ \text{ ask (x)} \Rightarrow \neg \text{ ask (shy)}$$

$$\bullet\!\!\!\circ \text{ give (shy)} \Rightarrow \diamond \text{ ask(shy)}$$

2 THE KNOWLEDGE SHARING EFFORT (KSE)

2.1 Introduction:

- Initiated by DARPA circa 1990
- Developing techniques, methodologies and software tools for *knowledge sharing* and *knowledge reuse*.
- KQML/KIF is probably the best-known ACL, currently undergoing standardisation. This is comprised of two parts:
  - the knowledge query and manipulation language (KQML); and
  - the knowledge interchange format (KIF).
- Large effort at defining common ontologies –software tools like `Ontolingua` for this purpose.
- KQML is an 'outer' language that defines various acceptable performatives:
  - `ask-if` ('is it true that…')
  - `perform` ('please perform the following action')
  - `tell` ('it is true that…')
  - `reply` ('the answer is …')
- KIF is a language for expressing message *content*.

- KIF is actually an ASCII representation of (extended) first-order logic, that looks like LISP

2.2 KQML

- *Specification of the KQML Agent-Communication Language plus example agent policies and architectures*, DARPA, External Interfaces Working Group, 1993.
- *A proposal for a new KQML Specification*, Yannis Labrou and Tim Finin, 1997.
- There are also many dialects and "extended" versions of KQML
- High-level. Message oriented, ***communication language*** and protocol for information exchange independent of content syntax and ontology.

Transport → ***Language*** → *Policy* (conversation protocol) → *Architecture*

- KQML is independent of
  - the transport mechanism (TCP/IP, email, CORBA, etc.)
  - the content language (KIF, SQL, STEP, Prolog, etc.)
  - the ontology assumed by the content
- Includes primitive message types of particular interest to building interesting agent architectures (e.g., for mediators, sharing intentions, etc.)
- Originally defined as a language with a particular syntax which is based on Lisp (Common Lisp Polish-prefix notation), because:
  - is readable by humans
  - simply for programs to parse, and
  - transportable by many inter-application messaging platforms
- Unlike Lisp function invocations, parameters in performatives are indexed by keywords and therefor independent.
- Alternate syntaxes have been used (e.g., SMTP, MIME, http, etc.)
- Proposals for a meta-syntax that can support different syntactic dialects.

2.3 KQML Semantics

- Semantic model underlying KQML is a simple, uniform context for agents to view each others' capabilities.
- Each agent appears, on the outside, as if it manages a knowledge base, so it is called *virtual knowledge base* (VKB), therefore communication is with regard to the VKB
- Classify the VKB statements can be classify into two categories:
  - *beliefs*: information about itself and its external environment, including the VKBs of other agents,
  - *goals*: states of its external environment that the agent will act to achieve

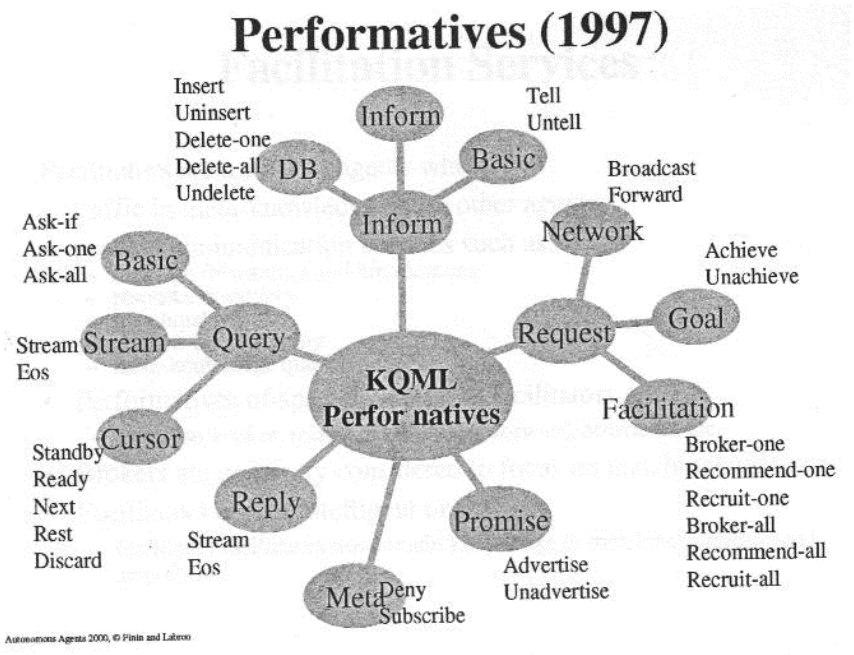
- Agents talk about the contents of their VKBs using KQML. But the encoding of statements in VKBs can use a variety of representation languages.
- The only restriction on such a representation is that it be sentential., so that expressions using that representation can be viewed as entries in a VKB, and that sentences have an encoding as ASCII string can be embedded in KQML messages.

2.4 KQML Messages

- Represents a single *speech act* or *performative*
  `ask, tell, reply, subscribe, achieve, monitor, …`
- With an associated *semantics and protocol*
  $tell(I, j, B_i\phi) = fp[B_iB_i\phi \wedge \ldots$
- And a list of *attribute/value pairs*
  `:content, :language, :from, :in-reply-to`

```
(tell      :sender         bhkAgent
           :receiver       fininBot
           :in-reply-to    id7.24.97.45391
           :ontology       ecbk12
           :language       Prolog
           :content        "price(ISBN3429459, 24.95)")
```

2.5 Reserved Performatives(1997)



2.6 Reserved Parameter Keywords(1997)

`:sender`	the actual sender of the performative
`:receiver`	the actual receiver of the performative
`:from`	the origin of the performative in *:content* when forward is used
`:to`	the final destination of the performative in *:content* when forward is used
`:in-reply-to`	the expected label in a response to a previous message (same as the *:reply-with* value of the previous message)
`:reply-with`	the expected label in a response to the current message
`:language`	the name of the representation language of the *:contenti* parameter
`:ontology`	the name of the ontology used in the *:content* parameter
`:content`	the information about which the performative expresses an attitude

2.7 KIF

- Extended version of first order predicate logic
- Simple list-based linear ASCII syntax. E.g.,
  (forall ?x (=> (P ?x) (Q ?x)))
  (exists ?person (mother mary ?person))
  (=> (apple ?x)(red ?x))
  (<<= (father ?x ?y) (and (child ?x ?y) (male ?x))
- KIF includes an axiomatic specification of large function and relation vocabulary and a vocabulary for numbers, sets and lists
- Distinguishes between implications and rules
  - Implication **=>** is a connective
  - Rules are directed (forward =>> or backward <<=)
  - Rules involve derivation
- KIF (but not ANSI KIF) allows non-monotonic rules via "circumscribing abnormality" using the *consis* predicate:
  - `(consis P)` is true if we are unable to derive (not P)
- Functions and relations are sets of lists in the universe of discourse.
  - They can be arguments to other functions & relations
  - They can be "applied" to arguments
- KIF can express knowledge about knowledge by allowing expressions to be treated as objects in the universe of discourse
  - KIF expressions are lists an can be referred to using the *quote* operator

2.8 Example of a KQML/KIF dialog

```
A to B:   (ask-if (> (size chip1) (size chip2)))
B to A:   (reply true)
B to A:   (inform (= (size chip1) 20))
B to A:   (inform (= (size chip2) 18))
```

3 THE FIPA ACL

3.1 Introduction to FIPA

- FIPA is the Foundation for Intelligent Physical Agents
- FIPA's purpose is to promote the success of emerging agent-based applications, services and equipment.
- FIPAS's goal is pursued by making available internationally agreed specifications that maximize interoperability across agent-based applications, services and equipment. The centrepiece of FIPA's work program is an ACL.
- FIPAS's works is built around annual rounds of FIPA specification deliverables:
- FIPA produces two kinds of specification:
  - **normative** specifications that mandate the external behaviour of an agent and ensure interoperability with other GIPA-specified subsystems;
  - **informative** specifications of applications for guidance to industry
- FIPA specifies the interfaces of the different components in the environment with which an agent can interact, i.e. humans, other agents, non-agent software and the physical world.

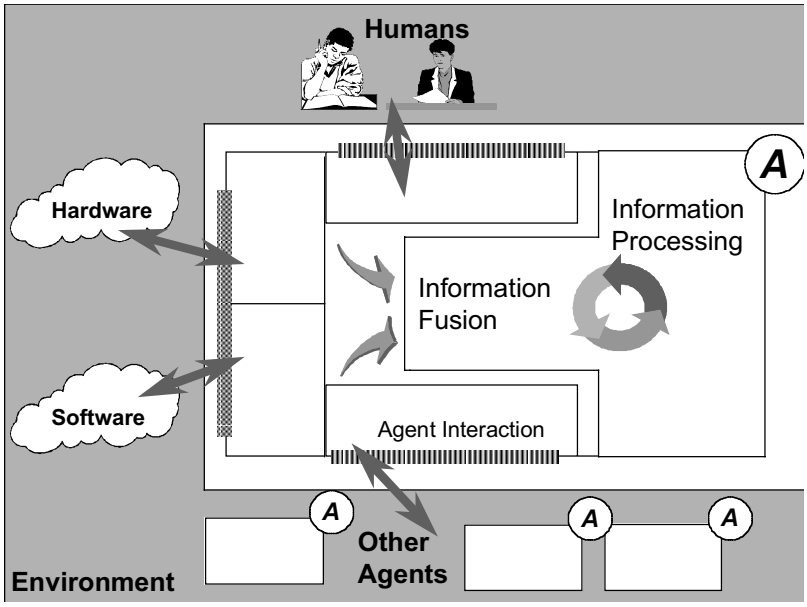


Figure 1 — Outline View of Agent Interaction

- FIPA97 laid the groundwork and focused on
  - Agent management (common components, agent lifecycle, platforms)
  - Agent communication (message format, semantics, interaction protocols)
  - Agent/Software interaction
  - Test Applications: Personal Travel Assistance, Personal Assistant, Audio/Video Entertainment & Broadcasting, and Network Management & provisioning
- FIPA 98 extended FIPA97, dealing with
  - Human-agent interaction
  - Agent mobility
  - Agent security
  - Ontology services
- FIPA 99 is work in progress:
  - TC1: Agent Management
  - TC2: Agent Communication Language
  - TC3: Agent/Software Interaction
  - TC4-TC7: Specification of Applications

3.2 The FIPA Agent Platform (Agent Management System)

- An Agent Platform provides an infrastructure in which agents can be deployed. An agent must be registered on a platform in order to interact with other agents on that platform or indeed other platforms.

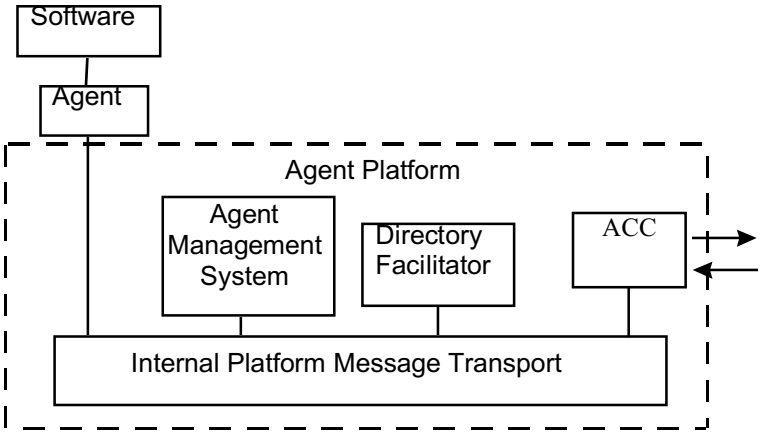


Figure 2 — Agent management reference model

- An AP consists of three capability sets ACC, AMS and default Directory Facilitator.
  - The AMS (Agent Management System) provides services like lifecycle management (creation, deletion, pausing, …), name registration, name lookup, and authentification
  - The DF (Directory Facilitator) provides yellow pages services which describe the attributes and capabilities of agents in the platform
  - The ACC (Agent Communication Channel) accepts and delivers message between agents on different platforms (+store and forward, +firewalls)

3.3 The FIPA Agent Communication Language

- Based on speech acts
- Messages are communicative acts (CAs) described in both a narrative form and a formal semantics based on modal logic
- Syntax is similar to KQML
- Specification provides a *normative* description of high-level interaction protocols (conversations)

3.4 The FIPA Agent/Software Integration

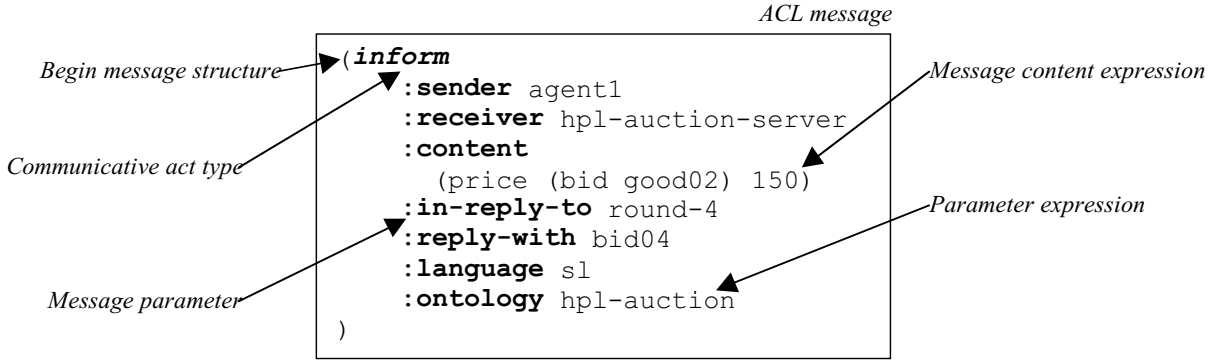
- Applies to any non-agentised software with which agents need to "connect".
- Such software includes legacy software, conventional database systems, middleware for all manners of interaction including hardware drivers.
- It suggests ways by which Agents may connect to software via "wrappers" including specifications of the wrapper ontology and the software dynamic registration mechanism.
- For this purpose, an *Agent Resource Broker* (ARB) service is defined which allows advertisement of non-agent services in the agent domain and management of their use by other agents.

3.5 FIPA Communicative acts by category

- Information exchange (requesting and passing)
- Negotiation
- Task delegation

Communicative act	Information passing	Requesting information	Negotiation	Action performing	Error handling
Accept-proposal			✓		
Agree				✓	
Cancel				✓	
Cfp			✓		
Confirm	✓				
Disconfirm	✓				
Failure					✓
Inform	✓				
Inform-if (macro act)	✓				
Inform-ref (macro act)	✓				
Not-understood					✓
Propose			✓		
Query-if		✓			
Query-ref		✓			
Refuse				✓	
Reject-proposal			✓		
Request				✓	
Request-when				✓	
Request-whenever				✓	
Subscribe		✓			

3.6 Message structure



3.7 Message parameters

Message Parameter:	Meaning:
`:sender`	Denotes the identity of the sender of the message, i.e. the name of the agent of the communicative act.
`:receiver`	Denotes the identity of the intended recipient of the message. Note that the recipient may be a single agent name, or a tuple of agent names
`:content`	Denotes the content of the message; equivalently denotes the object of the action.

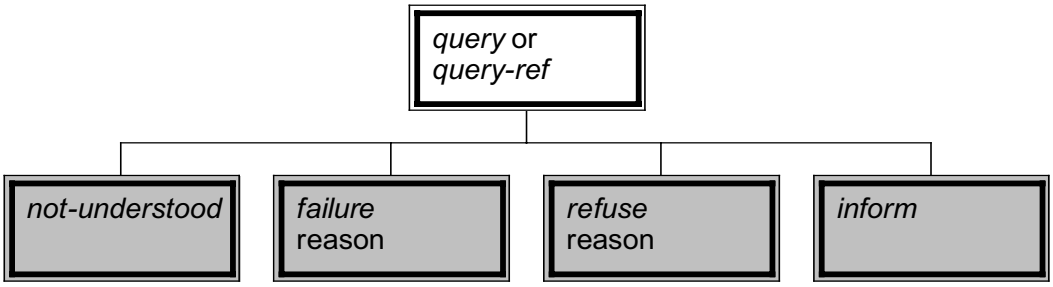
`:reply-with`	Introduces an *expression* which will be used by the agent responding to this message to identify the original message. Can be used to follow a conversation thread in a situation where multiple dialogues occur simultaneously.
	E.g. if agent i sends to agent j a message which contains
	`:reply-with query1,`
	agent j will respond with a message containing
	`:in-reply-to query1.`

`:in-reply-to`	Denotes an expression that references an earlier action to which this message is a reply.
`:envelope`	Denotes an expression that provides useful information about the message as seen by the message transport service. The content of this parameter is not defined in the specification, but may include time sent, time received, route, etc
`:language`	Denotes the encoding scheme of the content of the action.
`:ontology`	Denotes the ontology which is used to give a meaning to the symbols in the content expression.
`:reply-by`	Denotes a time and/or date expression which indicates a guideline on the latest time by which the sending agent would like a reply.
`:protocol`	Introduces an identifier which denotes the *protocol* which the sending agent is employing.
`:conversation-id`	Introduces an expression which is used to identify an ongoing sequence of communicative acts which together form a conversation. A conversation may be used by an agent to manage its communication strategies and activities. In addition the conversation may provide additional context for the interpretation of the meaning of a message.

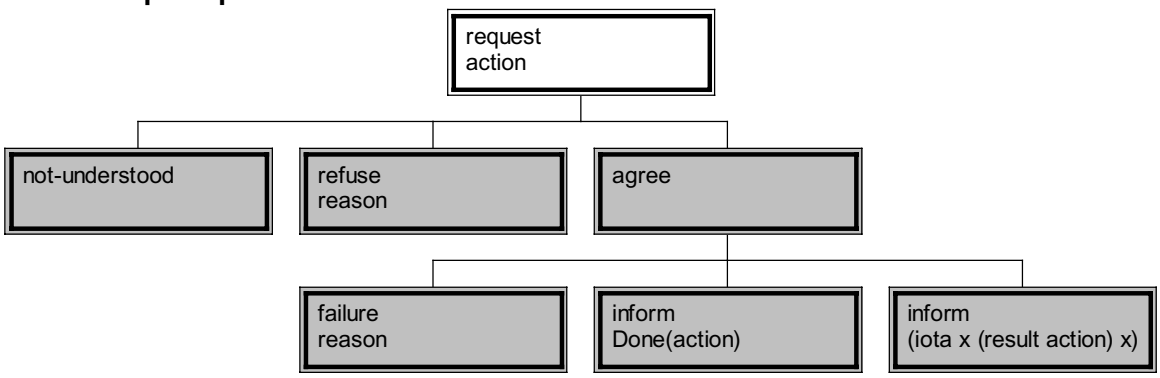
3.8 FIPA interaction protocols (IP)

- Interaction protocols define structured conversations
- Based on CAs
- Basis for dialogues between agents
- Set of pre-defined IPs
- Own IPs can be defined

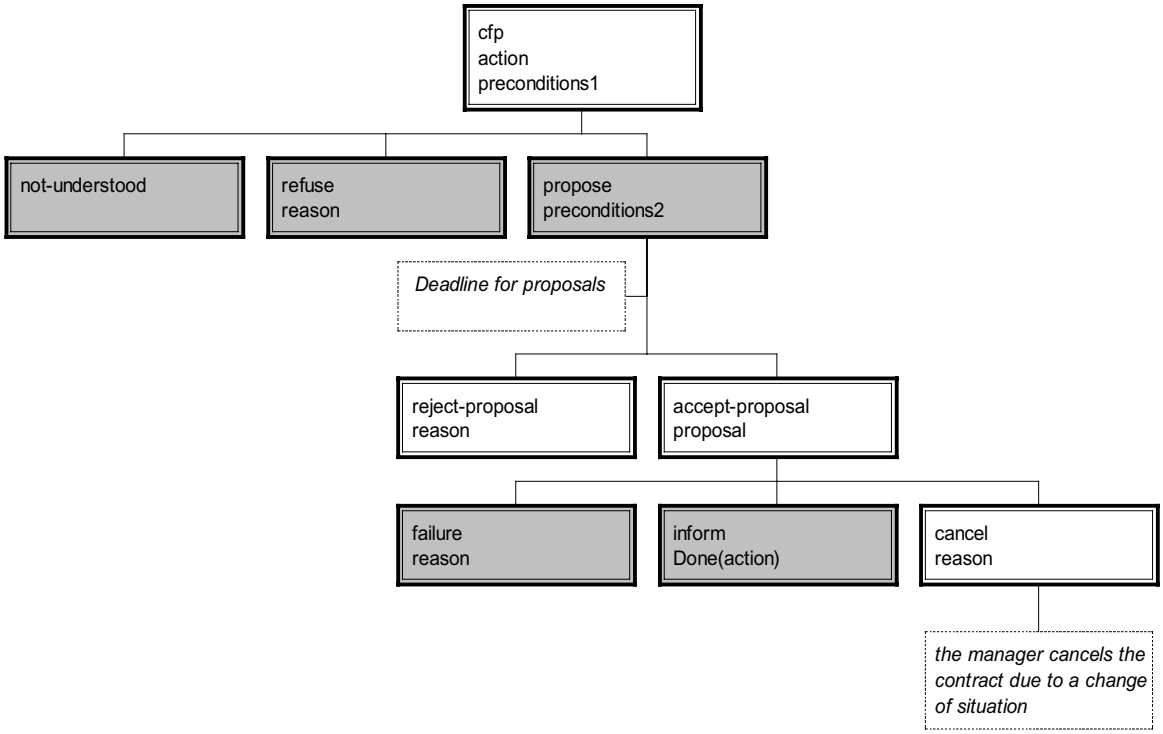
3.8.1 Query protocol



3.8.2 Request protocol



3.8.3 Contract-net protocol



4 Ontologies and FIPA

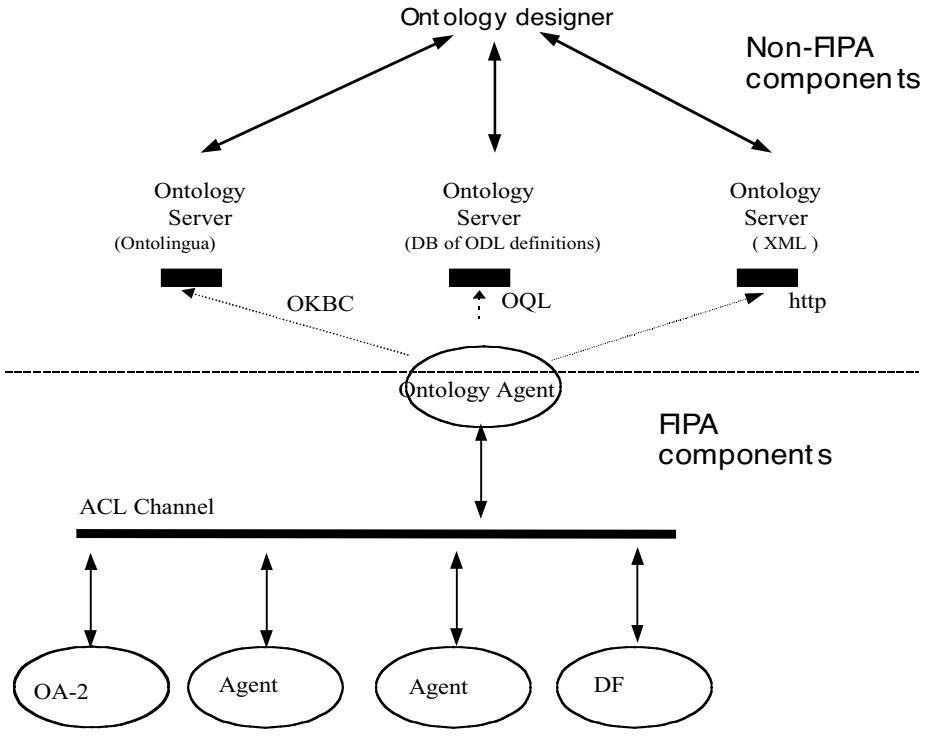


Figure 1 - Reference Model

- The Ontoloy Agents provide ontology related services to FIPA agents
  - Helps a FIPA agent in selecting a shared (sub) ontology for communication
  - Create and update an ontology
  - Translate expressions between different ontologies
  - Respond to query for relationhis between different terms of between ontologies
  - Discovery of public ontologies in order to access them
- FIPA specifies **fipa-meta-ontology** as the ontology used to talk about ontologies
- This is largely based on the OKBC model

5 ACL SEMANTICS

5.1 Introduction: agents theory

- Agent theory is needed to give a *semantics* to the architectures, languages and tools
- Without such a semantics. It is never clear exactly *what* is happening, or *why* it works.
- In agent-based systems, we have a bag of concepts and tools, which are intuitively easy to understand (by means of metaphor and analogy), but we need theory to reach any kind of profund understanding of these tools.
- Agent theorists start with the (strong) view of agents as intentional systems.
  - Attemps to produce a coherent account of how the components of an agent's cognitive state hold together: the theory of intention by Cohen & Levesque

5.2 Theories of attitudes

- First, we need to identify a tractable subset of attitudes, and a model of how they interact to generate system behaviour.
- Two categories (Bratman)
  - *Information attitudes*: belief, knowledge
  - *Pro-attitudes*: desire, intention, obligation, commitment, choice…
- Two aspects of logical formalism for intentional notions:
  - language of formulation (syntactic)
  - semantic model
- Two fundamental approaches to the syntactic problem
  - use a *modal language*
  - use a *meta-language*

5.3 Possible world semantics

- Logic of knowledge and belief is known as *epistemic* or *doxastic logic*

- Hintikka, 1962 → idea of applying the *possible world semantics* to intentional notions, using the Kripke structure as his logical tool.
- The language of possible worlds is a ***normal modal logic***
  - □ *necessity*: a proposition which is true in all worlds
  - ◊ *possibility*: a proposition that is true in at least on world
- *Epistemic logic* was developed by replacing the *necessity* operator with the (K) operator for *knowledge*.
- The worlds that an agent belief as possible as called *epistemic alternative*
- Semantics are defined in terms of a *model* or a *frame* $M = (W, R, \pi)$
  - *W* is a set of possible worlds;
  - *R* is a binary relation on *W*, called the *accesibility relation*;
  - π is an *assignment function* which determines for each world $w \in W$, the truth values of the atomic propositions in *w*
- Accessibility relation: if $(w, w') \in R$, then if an agent was actually in world *w*, as far as it was concerned, it might be in world *w'*
- Semantics of formulae are given relative to worlds:
  ϕ is true in world *w* iff ϕ is true in all worlds *w'* such that $(w, w') \in R$

  $(M, w) \models K_i\phi$ iff $\forall w' \in W$. if $(w, w') \in R_i$ then $(M, w') \models \phi$
- Axioms:

K		$\models K(\phi \Rightarrow \psi) \Rightarrow (K\phi \Rightarrow K\psi)$
T	(reflexive)	$K\phi \Rightarrow \phi$
D	(serial)	$K\phi \Rightarrow \neg K\neg\phi$
4	(transitive)	$K\phi \Rightarrow KK\phi$
5	(euclidean)	$\neg K\neg\phi \Rightarrow K\neg K\neg\phi$
NEC:		if $\models \phi$ then $K\phi$

- Imposing different restrictions on the accesibility relation, we obtain different systems of modal propositional logic:
  - KTD45 (S5) is the logic of *idealised knowledge*
  - KD45 (weak-S5) is the logic of *idealised belief*
- Interpretation of axioms

  K: An agent's knowledge is *closed under implication*, that is, it knows all the consequences of its beliefs

  T: The *knowledge axiom*: what is known is true

  D The *consistency axiom*: if you know ϕ, you can't also know $\neg\phi$

  4 *Positive introspection*: if you know ϕ, you know you know ϕ

  5 *Negative introspection*: you are aware of what you don't know
- K + NEC = *Logical omniscience* = An agent believes all valid formulae and knows all the logical consequences. This is an undesirable property !

- Despite this disadvantages, possible world semantics has been commonly used by the practitioners and still appears to be the most appropriate model for describing and reasoning about belief and knowledge.

5.4 Cohen & Levesque Theory of Rational Agency

- Intention = Choice + Commitment
- Intentions concerned with future
- Integration of Agent Theory and Semantics of Communication Primitives
- (BEL x p)
  - P follows from x's beliefs
- (GOAL x p)
- (BMB x y z) =
  (BEL x p) ∧
  (BEL x (BEL y p)) ∧
  (BEL x (BEL y (BEL x p)))
- Internal commitment (persistent goal)
  (P-GOAL x p q) =
  (BEL x ¬p) ∧
  (GOAL x (LATER p)) ∧
  [KNOW x (PRIOR [(BEL x p] ∨ (BEL x ¬p) ∨ (BEL x ¬q)
  ¬[GOAL x (LATER p)])]
- Intention
  (INTEND x a q) =
  (P-GOAL x [DONE x (BEL x (HAPPENS a))?;a] q)

5.5 FIPA ACL Semantics

- Management and facilitation primitives (register, broker, recruit, etc.) are not part of the ACL
- Primitives can be defined compositionally from "core" primitives
- Use of a powerful language to define agents' states: Semantic Language (SL)
- Semantics based on mental attitudes
- The meaning of primitives is given in terms of
  - *Feasibility Preconditions* (FPs): define the conditions of a CA that ought to be true before an agent may plan to execute the CA
  - *Rational Effect* (RE): the effect that an agent hopes to bring about by performing an action (but with no guarantee that the effect will be achieved)
- Semantics are normative, agents which claim to conform to the FIPA specification ACL must behave in accordance with the definitions herein.

5.6 Model of communicative acts

- Model that underlies the semantics of the FIPA ACL
- It is not a proposed architecture or a structural model of the agent design
- An agent plans to meet its goals ultimately by communicating with other agents. The agent will select acts based on the relevance of the acts' expected outcome or *rational effect* to its goals. However, it cannot assume that the rational effect will necessarily result form sending the messages.

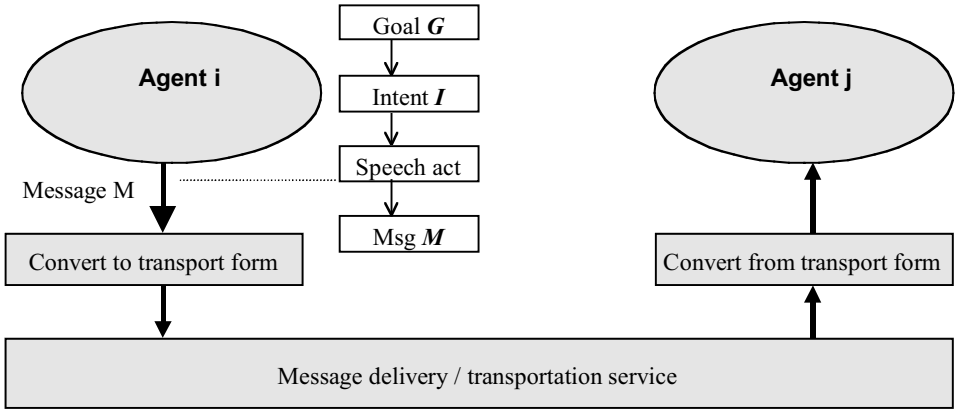


Figure 2 Message passing between two agents

5.7 The SL Language

- Used to define the semantics of FIPA ACL
- Logical propositions are expressed in a logic of mental attitudes and actions
- The logical framework is a first order modal language with identity (similar to C&L, see [Sadek 91a]).
- SL provides formalisation for three primitive mental states: *belief*, *uncertainty* and *choice* (or, to some extent, *goal*); *intention* is defined as a *persistent goal*.
  - $B_i p$ "*i* (implicitly) believes (that) *p*"
  - $U_i p$ "*i* is uncertain about *p* but thinks that *p* is more likely than ¬*p*"
  - $C_i p$ "*i* desires that *p* currently holds"
- To talk about complex *plans*, events (or actions) can be combined to form *action expressions*
  - $a_1; a_2$ is a *sequence* in which a_2 follows a_1
  - $a_1 \mid a_2$ is a non *deterministic choice*, in which either a_1 happens or a_2, but not both
- Operators to enable reasoning about actions
  - *Feasible*(*a*, *p*) means that *a* can take place and if it does *p* will be true just after that
  - *Done*(*a*, *p*) means that *a* has just taken place and *p* was true just before that

- *Agent*(*i*, *a*) means that *i* denotes the only agent performing, or that will be performing, the actions which appear in action expression *a*.
- *Single(a)* means that *a* denotes an action expression that is not a sequence. Any individual action is *Single*. The composite act `a ; b` is not *Single*. The composite act `a | b` is *Single* iff both a and b are *Single*.

- *Persitent goal* and *intention* are defined from belief, choice and events:
  - An agent *i* has *p* as a *persistent goal*, if *i* has *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. $PG_i p$ "*i* has *p* as a persistent goal"
  - *Intention* is defined as a persistent goal imposing the agent to act. $I_i p\ n$ "*i* has the intention to bring about *p*".

5.8 Underlying Semantic Model

- **Property 3**: If an agent has the intention that a communicative act be performed, it necessarily has the intention to bring about the rational effect of the act.

  $\models I_i \text{ Done(a)} \Rightarrow I_i \text{ RE}$

- **Property 4**: When an agent, it should believe that the agent performing the act has the intention to achieve the rational efect of the act. This is called the intentional effect.

  $\models B_i (\text{Done(a)} \wedge \text{Agent (j,a)} \Rightarrow I_i \text{ RE(a)}$

5.9 Primitive communicative acts

- **The assertive Inform:** the sender informs the receiver that a given proposition is true

  Formal model:

  <i, inform(j, ϕ)>

  FP: $B_i\phi \wedge \neg B_I(\text{Bif}_j\phi \vee \text{Uif}_j\phi)$

  RE: $B_j\phi$

  Example: Agent i informs agent j that (it is true that) it is raining today:

```
(inform
      :sender i
      :receiver j
      :content "weather (today, raining)"
      :language Prolog
      :ontology weather42)
```

- **The directive Request**: the sender requests the receiver to perform some action. One important class of uses if the request act is to request the receiver to perform another communicative act.

  Formal model:

  <i, request (j, a)>

FP: FP(a) [i\j] ∧ B_i Agent(j, a) ∧ B_i ¬PG_j Done(a)

RE: Done(a)

Example: Agent i requests j to open a file

```
(inform
      :sender i
      :receiver j
      :content "open \"db.txt\" for input"
      :language vb)
```

- **Confirming an uncertain proposition: Confirm**

  Formal model:

  <i, confirm(j, ϕ)>

  FP: B_iϕ ∧ $B_i$$U_j$ϕ

  RE: B_jϕ

  Example: Agent i confirms to agent j that it is, in fact, true that it is snowing today.

```
(confirm
   :sender i
   :receiver j
   :content "weather (today, snowing)"
   :language Prolog)
```

- **Disconfirming an uncertain proposition: Disconfirm**

  Formal model:

  <i, disconfirm(j, ϕ)>

  FP: B_i¬ϕ ∧ B_i (U_jϕ ∧ B_jϕ)

  RE: B_jϕ

  Example: Agent i, believing that agent j thinks that a shark is a mammal, attemps to change j's belief:

```
(disconfirm
   :sender i
   :receiver j
   :content (mammal shark))
```

5.10 Composite Communicative Acts

- Communicative acts defined by composition using the disjunction operator "|"
- **The inform-if act:** a macro action for the agent of the action to inform the recipient wheter or not a proposition is true.

  Formal model:

  <i, inform-if(j, ϕ)> ≡ <i, inform(j, ϕ)> | <i, inform(j, ¬ϕ)>

  FP: Bif_iϕ ∧ ¬B_i (Bif_jϕ ∧ Uif_jϕ)

  RE: Bif_jϕ

  Example: Agent i requests j to inform whether Lannion is in Normandy.

```
(request
  :sender i
  :receiver j
  :content
    (inform-if :sender j
               :receiver i
               :content "in( lannion, normandy )"
               :language Prolog)
  :language sl)
```

Agent j replies that it is not.

```
(inform :sender j
        :receiver i
        :content "\+ in( lannion, normandy )"
        :language Prolog)
```

- **The query-if act:** the action of asking another agent wether or not a given proposition is true

Formal model:

$\langle i, \text{query-if}(j, \phi) \equiv \langle i, \text{request}(j, \langle j, \text{inform-if}(i, \phi)\rangle)\rangle$

FP: $\neg Bif_i\phi \wedge \neg Uif_i\phi \wedge B_i \neg PG_j \text{Done}(\langle j, \text{inform}(i, \phi)\rangle)$

RE: $\text{Done}(\langle j, \text{inform}(i, \phi)\rangle \mid \langle j, \text{Inform } i, \neg\phi\rangle)$

Example: agent i asks agent j if is registered with domain server d1

```
(query-if
  :sender i
  :receiver j
  :content
     (registered (server d1) (agent j))
  :reply-with r09
  ...)
```

Agent j replies that it is not:

```
(inform
  :sender j
  :receiver i
  :content (not (registered (server d1) (agent j)))
  :in-reply-to r09)
```

6 EVALUATION OF ACLs AND SEMANTIC APPROACHES

6.1 Introduction

- Different ACLs: Different semantic approaches
  - KQML semantics (Labrou, 1996)
  - FIPA ACL (FIPA ACL specification, 1997, 1999)
  - ACL semantics (Cohen & Levesque)

6.2 How do KQML and ACL differ?

- Different semantics: mapping of KQML performatives to FIPA performatives and viceversa is a futile exercise
- Different treatment of the "administration primitives"; in FIPA ACL register, unregister, etc. are treated as requests for action with reserved (natural language) meaning
- No "facilitation primitives", e.g., broke, recommend, recruit, etc., in FIPA ACL
- Reserved content language: a very murky issue …
  - Some FIPA messages (e.g., request) use SL as their content language. An agent that observe such messages have to "understand" some SK; how much depends on the particular message

6.3 Which ACL should I use? Which one is better?

(Finin & Labrou, Autonomous Agents 2000, Tutorial on ACLs: past, present and future)

- The "sad truth" is that programmers do not care about semantics and their details.
- As long as the agent dos not implement modalities (belief, intention, etc.) the semantic differences are irrelevant to the developer
- The similar syntax guarantees that a developer will not have to alter the code that receives, parses and send messages
- The code that processes the primitives should change depending on whether the code observes the proper semantics.
- FIPA ACL is more powerful with composing new primitives. This is due to the power of the SL language as a content language to describe agents' states.
- KQML's weakness is it religious on-commitment to a content language.
- Both have shortcomings; there are features that developers would like to see in an ACL.

6.4 Shortcomings of currents ACLs

- Intentional level description; which mental attitudes, what definitions?
- Problems with mental attitudes: from theory to practice
- Can all desirable communication primitives be modeled as speech acts? Should they?
- How can we test an agent's compliance with the ACL?
- Ease of extending an ACL

6.5 Are there any other ACLs?

- KQML and FIPA ACL are the only fully specified ACLs
- There exists many variants of KQML that do not qualify KQML as a fully specified ACL

- FIPA ACL only exists in paper; its specification has not been challenged yet,partly due to the lack of applications using it
- Two important KQML variants exist; both challenge some fundamental KQML concepts:
  - KQML + KIF, from Stanford (Genesereth)
  - Work by Cohen, Levesque and Smith

6.6 Alternatives to ACLs

- There are many alternatives to using ACLs for communicating and sharing information.
- From oldest to newest:
  - *Natural language*:
  - *Database languages* (SQL)
  - *Domain dependant* (EDI)
  - *Distributed object systems* (CORBA)
  - OKBC
  - *Service languages* (e-speak, BizTalk)
  - *Web languages* (XML, RDF, DAML)
- One size won't fit all, so we need to appreciate the strengths and weaknesses.
- We can also consider mixing, matching and morphing

7 FIPA APPLICATIONS. THE PERSONAL ASSISTANT

7.1 FIPA Applications

- *User Assistant Applications*:
- *Information Retrieval Applications*
- Entertainment Applications
- Service Management Applications
- Business Management Applications
- Manufacturing Management Applications
- Service Robotics Applications
- Cooperative Tasks Management Applications
- Research Applications

7.2 Overview of the Personal Assistant

- A personal assistant (PA) is a software agent that acts semi-autonomously for and on behalf of a user, modelling the interests of the user and providing services to the user or other people and PAs as and when required.

- These services include managing a user's diary, filtering and sorting e-mail, managing the user's activities, locating and delivering (multimedia) information, and planning entertainment and travel.
- It is like a secretary, it accomplishes routine support tasks to allow the user to concentrate on the real job, it is unobtrusive but ready when needed, rich in knowledge about user and work.
- Some of the services may be provided by other agents (e.g. the PTA) or systems, the Personal Assistant acts as an interface between the user and these systems.
- In the FIPA'97 test application, a Personal Assistant offers the user a unified, intelligent interface to the management of his personal meeting schedule. The PA is capable of setting up meetings with several participants, possibly involving travel for some of them. In this way FIPA is opening up a road for adding interoperability and agent capabilities to the already established domain of Personal Information Management

7.3 Reference Model

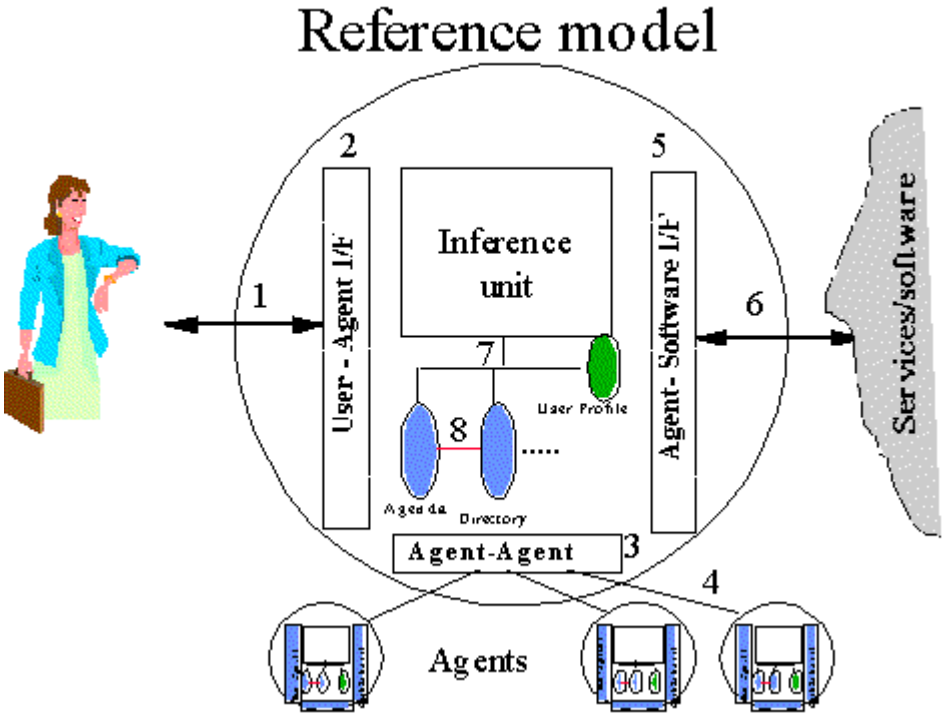


Figure 3 — Personal Assistant Reference Model

- The reference model includes the following interfaces/protocols of interaction that are candidates for standardisation.
  - User-Agent Dialogue
  - Multi-Modal User-Agent Interface
  - Agent-Agent Communication Interfaces & Protocols for A-A Interaction
  - Agent-Software Interfaces & A-S Communication Protocols
  - Agent-Functions Interfaces
- Concrete examples of the PA's functions/services needed
  - Directory Services
  - Meeting Scheduling Services
  - Information Management Services
  - Travel Planning Service
- The chosen scenario is that of arranging meetings among several participants, located across companies and using different calendar management systems.
- The selected service is an integration of meeting scheduling and travel assistance. The user asks the agent to set up a meeting with several participants. Because the meeting may involve travelling for some of the participants, travel planning forms part of the meeting scheduling.

7.4 System Architecture

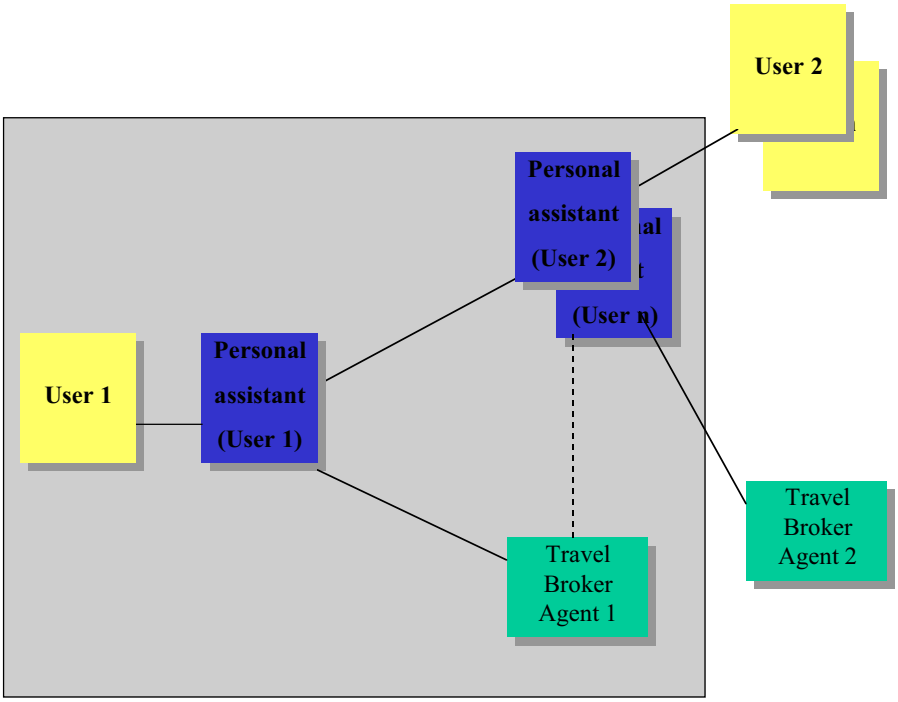


Figure 4 — Agent Interactions in Personal Assistant Application Scenario

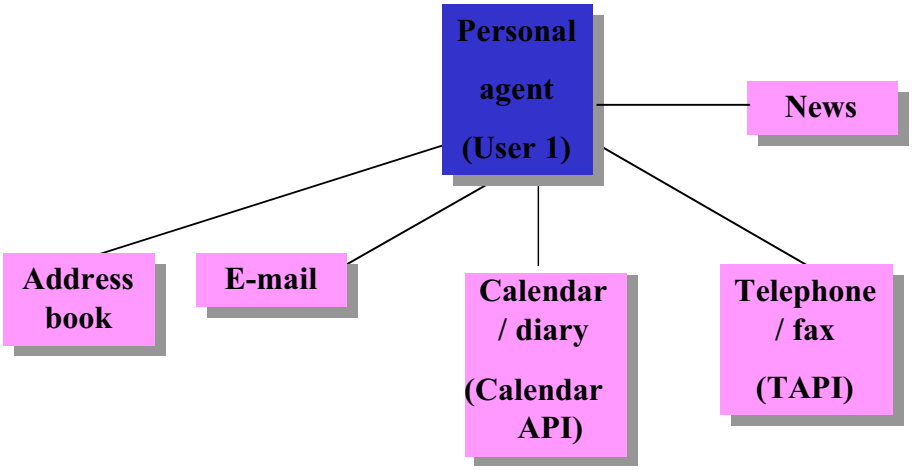


Figure 5 — Integration of External Software

7.5 FIPA technologies used

8 Agent Management

- This application makes use of the Directory Facilitator and, as such, requires agents to register and deregister with the DF.
- Optional attributes: search

• Agent Communication Language

- CAs: `cfp, accept-proposal, reject-proposal, not-understood, propose, refuse, inform, failure, perform`
- Interaction protocols: FIPA-Cotract-Net and FIPA-Request

• Human-Agent Interaction (*informative*)

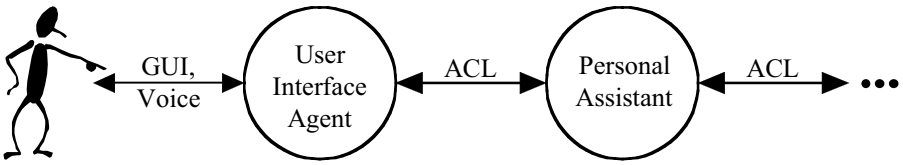


Figure 6 — Agent-Human Interaction via User Interface Agent

• Contents of Interaction (*informative*)

a) U -> PA : give task to arrange meeting (including reporting requirements)
b) PA -> U: progress status of task, at least success or failure (with reasons)
c) U -> PA: permission to commit to meeting
d) PA->U: request for permission to commit
e) PA -> U: inform commitment made
f) U -> PA: degree of delegation authority

- Agent / Software Integration
  - diary or calendar management system
  - address book
  - email/FAX
  - TAPI
  - Personal Travel Assistance

7.6 The PA Content Language

- The primary requirement if the content language is the representation of meetings, trips, the actions of scheduling the meeting (as carried out by the initiator's PA), and the action of participating in the meeting(as carried out by the human participants). Due to its simplicity , standard *s-expression* syntax is chosen.
- The syntax is expressed in standard EBNF format as summarised in [FIPA Document fipa7612.doc].
- Expressions in the content language appear in the ACL message format syntax as value expressions of the "`:content`" parameter in the ACL message syntax, specifically as in: :`content` "(" PA-content-message Proposition ")" .

```
PA-content-message      =     "(" PA-Action ")".
PA-Action               =     PA-Meet | PA-Travel | PA-Schedule .
PA-Meet                 =     "(PA-Meet" ObjectId PA-Meeting Result
                              Status Agent StartTime Duration Deadline
                              ")" .
PA-Travel               =     "(PA-Travel" ObjectId PA-Trip Status Agent
                              StartTime Duration Deadline ")" .
PA-Schedule             =     "(PA-Schedule" ObjectId PA-Object Result
                              Status Agent StartTime Duration Deadline
                              ")" .
PA-Object               =     PA-Meeting | PA-Trip .
PA-Meeting              =     "(PA-Meeting" ObjectId Initiator
                              vCalendarObject Protocol ")" .
```

7.7 Interaction protocols

- **Negotiating Meeting Details (normative):**the accept-proporal is used only if all bids (have a time frame in common.

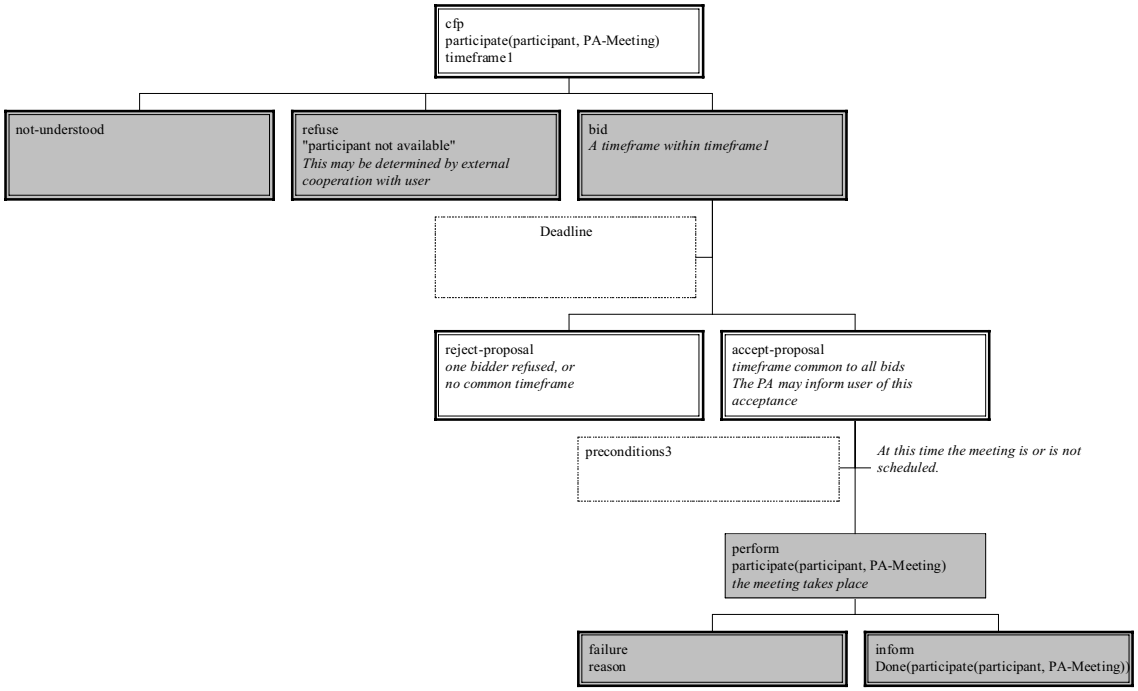


Figure 7 — FIPA-ContractNet (applied to meeting negotiation)

- **Scheduling a meeting (informative):** an Order protocol can be used in order to initiate negotiation of a meeting

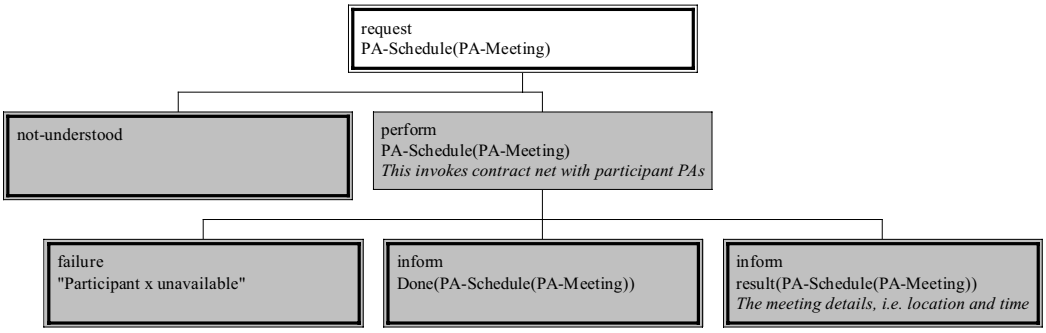


Figure 8 — PA-Order (applied to meeting scheduling) (informative)

- **Confirmation with user (informative):** the FIPA-RequestWhen protocol can be used by the PA to confirm the availability of the user at a suggested meeting time.

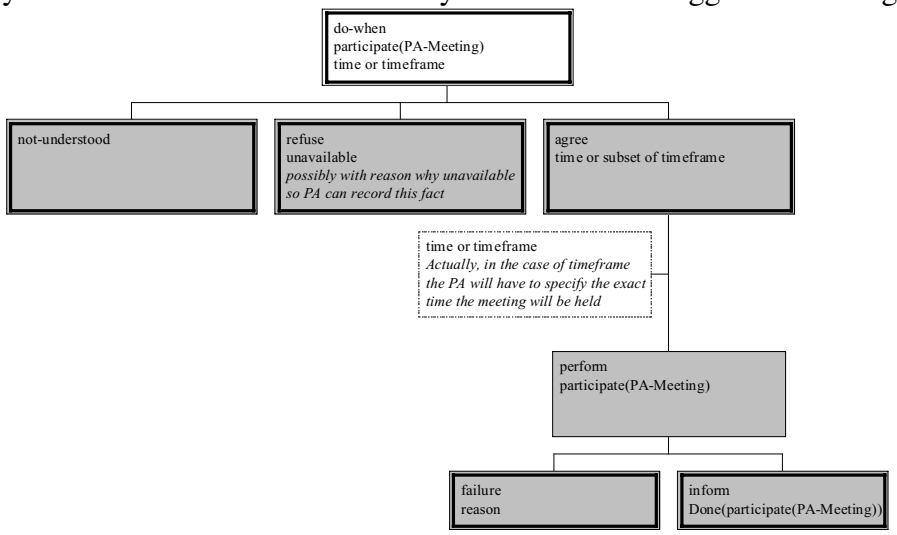


Figure 9 — FIPA-RequestWhen (applied to meeting scheduling) (informative)

7.8 Example

- Suppose John Doe wants to schedule an hour long meeting with some colleagues during some time on a given day. Then John Doe's PA will send the following message to the personal agents of the desired participants:

```
(cfp
    :sender UA-Donald
    :receiver UA-Wiet
    :content ((PA-Meet
                  :ObjectID WietMeet123
                  :Agent Hans Mustermann
                  :PA-Meeting (BEGIN:VCALENDAR
VERSION: 1.0
BEGIN:VEVENT
UID: 123
SUMMARY: FIPA Demo
ATTENDEE:Hans Mustermann
ATTENDEE;ROLE=ORGANIZER:John Doe
LOCATION:Siemens MchP 53.512
CATEGORIES: X-FIPA-Test
DESCRIPTION: This is just a test meeting. Please do not attend it.
SEQUENCE: 0
PRIORITY: 0
DTSTART: X-FIPA-UnderNegotiation
DTEND: X-FIPA-UnderNegotiation
STATUS: UNDER NEGOTIATION
X-FIPA-ORGANIZER: John Doe
END:VEVENT
END:VCALENDAR
                                  )
```

```
              )
             (InTimeIntervals(WietMeet123.PA-
Meeting.DTSTART,[[ 1200,1800]]) ∧
DTEND = DTSTART + 60 )
              )
    :ontology FIPA-PA
    :conversation-id UA-Donald345
    :protocol FIPA-ContractNet
    :reply-with Response123
    :reply-by 101097T1300
)
```

7.9 FIPA platforms

- Mecca (Siemens)
- Jade (CSELT) http://drogo.cselt.stet.it/ufv/jade/JADEFeatures.html
- FIPA-OS (Nortel)
- FIPA-Smart http://labs.bt.com/projects/agents/zeus/
- CoABS (GITI)

8 REFERENCES AND LINKS

9 M. Woolridge, 2000, Multi-Agent Systems, *Seminario UPC*

10 Agent Communication Languages. Past, Present and Future, *Autonomous Agents 2000, Tutorial 9*

11 FIPA 97 specifications
- Part 1: Agent Management
- Part 2: Agent Communication Language
- Part 5: Personal Assistant

- FIPA 98 specifications
  - Part 12: Ontology Service
  - Part 13: Developers Guide
- A. Haddadi. Communication and Cooperation in Agent Systems. A Pragmatic Theory. *Lecture Notes in Artificial Intelligence*
- M. R. Genesereth & S.P. Ketchpel. Software Agents. *Communications of the ACM, July 1994/Vol 37, No. 7*
- Knowledge Interchange Format. *Draft proposed American National Standard*
- DRAFT Specification of the KQML Agent-Communication Language.

- General information on software agents
  http://agents.umbc.edu
- The FIPA home
  http://www.fipa.org
- Information on KQML, KIF and ontologies
  http://www.agents.umbc.edu/kqml
  http://wwwagents.cs.umbc.edu/kif
- Information on Agent Communication Languages
  http://www.agents.umbc.edu/acl/