FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA 98 Specification

Part 8, Version 1.0

Human-Agent Interaction

Obsolete

Publication date: 23rd October 1998

Geneva, Switzerland

This is one part of the first version of the FIPA 98 Specification as released in October 1998.
The latest version of this document may be found on the FIPA web site:
http://www.fipa.org
Comments and questions regarding this document and the specifications therein should be addressed to:
fipa98@fipa.org
It is planned to introduce a web-based mechanism for submitting comments to the specifications. Please refer to the web site for FIPA's latest policy and procedure for dealing with issues regarding the specification.

Table of Contents

Foreword

The Foundation for Intelligent Physical Agents (FIPA) is a non-profit association registered in Geneva, Switzerland. FIPA's purpose is to promote the success of emerging agent-based applications, services and equipment. This goal is pursued by making available in a timely manner, internationally agreed specifications that maximise interoperability across agent-based applications, services and equipment. This is realised through the open international collaboration of member organisations, which are companies and universities active in the agent field. FIPA intends to make the results of its activities available to all interested parties and to contribute the results of its activities to appropriate formal standards bodies.

This specification has been developed through direct involvement of the FIPA membership. The 48 members of FIPA (October 1998) represent 13 countries world-wide.

Membership in FIPA is open to any corporation and individual firm, partnership, governmental body or international organisation without restriction. By joining FIPA each member declares himself individually and collectively committed to open competition in the development of agent-based applications, services and equipment. Associate Member status is usually chosen by those entities who want to be members of FIPA without using the right to influence the precise content of the specifications through voting.

The members are not restricted in any way from designing, developing, marketing and/or procuring agent-based applications, services and equipment. Members are not bound to implement or use specific agent-based standards, recommendations and FIPA specifications by virtue of their participation in FIPA.

This specification is published as FIPA 98 specifications ver 1.0. All these parts have undergone an intense review by members as well as non-members during the past year as preliminary versions have been available on the FIPA web site. FIPA members as well as many non-members have been conducting validation trials of the FIPA 97 specification during 1998 and will continue to subject the new output to further validation during the coming months. During 1999 FIPA will publish revised versions of the current specifications and is also planning to continue work on further specifications of agent based technology.

Introduction

The FIPA specifications represent the primary output of FIPA. It is important to appreciate that these specifications have been derived from examining requirements on agent technology posed by specific industrial applications chosen by FIPA so far, and described in Parts 4, 5, 6, and 7 of the FIPA 97 specifications.

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. FIPA produces two kinds of specifications:

normative specifications mandating the external behavior of an agent and ensuring interoperability with other FIPA-specified subsystems;

informative specifications of applications providing guidance to industry on the use of FIPA technologies.

In October 1997, FIPA released its first set of specifications, called FIPA 97, Version 1.0. During 1998, comments on this specification were received. Based upon these comments, parts of FIPA 97 were superseded by a second version released in October 1998, introducing minor changes only.

Furthermore, in October 1998 FIPA released a new set of specifications, called FIPA 98, version 1.0, of which this document is a part.

The following tables provide an overview of the complete set of FIPA specifications.

Sorted by part:

		Released October 1997	***Released October 1998***	
Part		**FIPA 97 Version 1.0**	**FIPA 97 Version 2.0**	**FIPA 98 Version 1.0**
1	N	*Agent Management*	Agent Management	Agent Management Extensions
2	N	*ACL*	ACL	
3	N	Agent Software Integration		
4	I	Personal Travel Assistant		
5	I	Personal Assistant		
6	I	Audio Visual Entertainment & Broadcasting		
7	I	Network Management & Provision		
8	N			Human-Agent Interaction
10	N			Agent Security Management
11	N			Agent Management Support for Mobility
12	N			Ontology Service
13	I/M			Developer's Guide

N == normative; I == informative; M == methodology; *Italicised == superseded*

Sorted by topic:

Topic	**FIPA 97***(Version 1.0, unless otherwise indicated)*	**FIPA 98** Version 1.0
Agent Management	1. Basic System *(Version 2.0)*	1. Extension to Basic System 10. Agent Security Management 11. Agent Management Support for Mobility
Agent Communication	2. Agent Communication Language *(Version 2.0)*	8. Human-Agent Interaction 12. Ontology Service
Agent S/W Integration	3. Agent Software Integration	
Reference Applications	4. Personal Travel Assistant 5. Personal Assistant 6. Audio/Visual Entertainment & Broadcasting 7. Network Management & Provisioning	

The parts of the FIPA 98 specifications are briefly described below.

Part 1 - Agent Management

This part covers agent management for inter-operable agents, and is thus primarily concerned with defining open standard interfaces for accessing agent management services. It also specifies an agent management ontology and agent platform message transport. This specification incorporates and further enhances the FIPA 97, Part 1, Version 2.0 specification. The internal design and implementation of intelligent agents and agent management infrastructure is not mandated by FIPA and is outside the scope of this part.

Part 8 – Human-Agent Interaction

This part deals with the human-agent interaction part of an agent system. It specifies two agent services: User Dialog Management Service (UDMS) and User Personalization Service (UPS). A UDMS wraps many types of software components for user interfaces allowing for ACL level of interaction between agents and human users. A UPS can maintain user models and supports their construction by either accepting explicit information about the user or by learning from observations of user behavior.

Part 10 – Agent Security Management

Security risks exist throughout agent management: during registration, agent-agent interaction, agent configuration, agent-agent platform interaction, user-agent interaction and agent mobility. The Security Management specification identifies the key security threats in agent management and specifies facilities for securing agent-agent communication via the FIPA agent platform. This specification represents the minimal set of technologies required and is complementary to the existing FIPA 97 and FIPA 98, Part 1 specifications. This part does not mandate every FIPA-compliant agent platform to support agent security management.

Part 11 – Agent Management Support for Mobility

This specification represents a normative framework for supporting software agent mobility using the FIPA agent platform. This framework represents the minimal set of technologies required and is complementary to the existing FIPA 97 and FIPA 98, Part 1 specifications. Wherever possible, it refers to existing standards in this area. The framework supports additional non-mobile agent management operations such as agent configuration. The specification does not mandate that every FIPA-compliant agent platform must support agent mobility, nor does it cover the specific requirements for agents on mobile devices with intermittent connectivity, which is covered by the scope of the existing FIPA Agent Management activity.

Part 12 – Ontology Service

This part deals with technologies enabling agents to manage explicit, declaratively represented ontologies. It specifies an ontology service provided to a community of agents by a dedicated Ontology Agent. It allows for discovering public ontologies in order to access and maintain them; translating expressions between different ontologies and/or different content languages; responding to queries for relationships between terms or between ontologies; and, facilitating identification of a shared ontology for communication between two agents.

The specification deals only with the communicative interface to such a service while internal implementation and capabilities are left to developers. The interaction protocols, communicative acts and, in general, the vocabulary that agents must adopt when using this service are defined. The specification does not mandate the storage format of ontologies, but only the way the ontology service is accessed. However, in order to specify the service, an explicit representation formalism, or meta-ontology, has been specified allowing communication of knowledge between agents.

Part 13 – FIPA **97 Developer's Guide**

The Developer’s Guide is meant to be a companion document to the FIPA 97 specifications, and is intended to clarify areas of specific interest and potential confusion. Such areas include issues that span more than one of the normative parts of FIPA 97.

1. Scope

This document forms part of the FIPA 1998 standard. It provides a specification dealing with technologies that support design or implementation of the human-agent interaction part of an agent system. This part of FIPA 98 defines basic functionality that can be utilized by an agent-based application which needs to interact with human users. The purpose of this document is to standardize the basic functionality and interface of agents that are able to (1) manage dialogs with users at a higher level of operations or to (2) support personalization of both human-agent interaction and other agent behavior by constructing and maintaining user models. User models may be user profiles, i.e. stores of explicitly given information about the user, or may be acquired by learning from observations of user behavior.

The specification defines a reference model, identifying necessary agent functionalities, messages and actions which define each of these functionalities, and objects that are used as action parameters. It builds upon FIPA 97 specifications and other parts of FIPA 98 specifications.

2. Normative reference(s)

FIPA 97 – International standard for the inter-operation of software agents – Part 1: Agent Management.

FIPA 97 – International standard for the inter-operation of software agents – Part 2: Agent Communication Language.

FIPA 97 – International standard for the inter-operation of software agents – Part 3: Agent/Software Integration.

P3P – Platform for Privacy Preferences (P3P) Syntax Specification. W3C Working Draft 2-July-1998. http://www.w3.org/TR/WD-P3P10-syntax

vCard – Internet Mail Consortium, "vCard – The Electronic Business Card Version 2.1", http://www.imc.org/pdi/vcard-21.txt

3. Terms and definitions

For the purposes of this specification, the following terms and definitions apply:

Action
A basic construct which represents some activity which an agent may perform. A special class of actions is the communicative acts.

Agent
An Agent is the fundamental actor in a domain. It combines one or more service capabilities into a unified and integrated execution model which can include access to external software, human users and communication facilities.

Agent cloning
The process by which an agent creates a copy of itself on an agent platform.

Agent code
The set of instructions used by an agent.

Agent Communication Language (ACL)
A language with precisely defined syntax, semantics and pragmatics that is the basis of communication between independently designed and developed software agents. ACL is the primary subject of this part of the FIPA specification.

Agent Communication Channel (ACC)
The Agent Communication Channel is an agent which uses information provided by the Agent Management System to route messages between agents within the platform and to agents resident on other platforms.

Agent data
Any data associated with an agent.

Agent invocation
The process by which an agent can create another instance of an agent on an agent platform.

Agent Management System (AMS)
The Agent Management System is an agent which manages the creation, deletion, suspension, resumption, authentication and migration of agents on the agent platform and provides a „white pages“ directory service for all agents resident on an agent platform. It stores the mapping between globally unique agent names (or GUID) and local transport addresses used by the platform.

Agent Platform (AP)
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. An AP consists of three capability sets ACC, AMS and default Directory Facilitator.

Agent Platform Security Manager (APSM)
An Agent Platform Security Manager is responsible for maintaining the agent platform security policy. The APSM is responsible for providing transport-level security and creating agent audit logs. The APSM negotiates the requested intra- and inter-domain security services of other APSM's in concert with the implemented distributed computing architectures, such as CORBA, COM, DCE, on behalf of an agent in its domain.

ARB Agent
An agent which provides the Agent Resource Broker (ARB) service. There must be at least one such an agent in each Agent Platform in order to allow the sharing of non-agent services.

Communicative Act (CA)
A special class of actions that correspond to the basic building blocks of dialogue between agents. A communicative act has a well-defined, declarative meaning independent of the content of any given act. CA's are modeled on speech act theory. Pragmatically, CA's are performed by an agent sending a message to another agent, using the message format described in FIPA 97, part 2.

Content
That part of a communicative act which represents the domain dependent component of the communication. Note that "the content of a message" does not refer to "everything within the message, including the delimiters", as it does in some languages, but rather specifically to the domain specific component. In the ACL semantic model, a content expression may be composed from propositions, actions, or other expressions.

Content Language
The *content* of a FIPA message refers to whatever the communicative act applies to. If, in general terms, the communicative act is considered as a sentence, the content is the grammatical object of the sentence. This content can be encoded in any language, the *content language*, denoted by the `:language` parameter of the communicative act.

Conversation
An ongoing sequence of communicative acts exchanged between two (or more) agents relating to some ongoing topic of discourse. A conversation may (perhaps implicitly) accumulate context which is used to determine the meaning of later messages in the conversation.

CORBA:
Common Object Request Broker Architecture, an established standard allowing object-oriented distributed systems to communicate through the remote invocation of object methods.

Directory Facilitator (DF)
The Directory facilitator is an agent which provides a „yellow pages“ directory service for the agents. It store descriptions of the agents and the services they offer.

Explicit & Implicit
An ontology is *explicit* when it is specified in declarative form as a set of axioms and definitions (e.g. as a set of Ontolingua statements) that an agent can refer to (e.g. by means of an OKBC interface). An ontology is *implicit*, when the assumptions on the meaning of its vocabulary are only implicitly embedded in some piece of software.

Feasibility Precondition (FP)
The conditions (i.e. one or more propositions) which need be true before an agent can (plan to) execute an action.

Illocutionary effect
See speech act theory.

Knowledge model
It is a specification of the set of primitives used by a certain class of representation languages. As such, a knowledge model can be considered as a meta-ontology. For instance, several ontology servers use an object oriented model of knowledge based on primitive notions like classes, frames, properties, constraints, axioms and functions. FIPA adopts for the specification of these notions the OKBC version 2.0.4 Knowledge Model, which is called FIPA-meta-ontology or FIPA knowledge model.

Knowledge Querying and Manipulation Language (KQML)
A de facto (but widely used) specification of a language for inter-agent communication. In practice, several implementations and variations exist.

Local Agent Platform
The Local Agent Platform is the AP to which an agent is attached and which represents an ultimate destination for messages directed to that agent.

Message
An individual unit of communication between two or more agents. A message corresponds to a communicative act, in the sense that a message encodes the communicative act for reliable transmission between agents. Note that communicative acts can be recursively composed, so while the outermost act is directly encoded by the message, taken as a whole a given message may represent multiple individual communicative acts.

Message content
See content.

Message transport service
The message transport service is an abstract service provided by the agent management platform to which the agent is (currently) attached. The message transport service provides for the reliable and timely delivery of messages to their destination agents, and also provides a mapping from agent logical names to physical transport addresses.

Meta-ontology
For allowing a FIPA agent to communicate through ACL messages about ontologies, it is necessary to describe the concepts used to speak about an ontology. This description is called the meta-ontology. It is an ontology itself as it provides the ontology to refer to another ontology. Therefore, the meta-ontology should be powerful enough to deal with all potentially available ontologies and make explicit, at least informally, these concepts.

Mobile agent
An agent that is not reliant upon the agent platform where it began executing and can subsequently transport itself between agent platforms.

Mobility
The property or characteristic of an agent that allows it to travel between agent platforms.

Ontology
An ontology gives meanings to symbols and expressions within a given domain language. In order for a message from one agent to be properly understood by another, the agents must ascribe the same meaning to the constants used in the message. The ontology performs the function of mapping a given constant to some well-understood meaning. For a given domain, the ontology may be an explicit construct or implicitly encoded with the implementation of the agent.

Ontology Agent
An agent that provides the Ontology Service specified in this specification. The main objective of the Ontology Agent is to offer to FIPA agents a unified view of the services offered by the different ontology servers. Its second objective is to allow an ontology server to be known by FIPA agents. Moreover some ontology agents can provide the agents

with services such as translation facilities. Like any other FIPA agent, the ontology agent has to be registered to the DF and to provide the DF with the published ontologies and available services.

Ontology Name
The ontologies referred to by the agents can be provided by different ontology servers. Consequently, these ontology names are constructed from: the OA name, and the ontology logical name (given by the ontology designer e.g. “car “).

Ontology Server
Provider of an Ontology Service, not necessarily in the FIPA domain, or FIPA-compliant. Examples of ontology servers already existing outside FIPA are: Ontolingua, XML/RDF ontology servers, ODL databases ontologies servers. Access to the services provided by these ontologies servers are based on various APIs such as the OKBC interface, the ODL interface or HTTP.

Ontology sharing problem
The problem of ensuring that two agents who wish to converse do, in fact, share a common ontology for the domain of discourse. Minimally, agents should be able to discover whether or not they share a mutual understanding of the domain constants.

Perlocutionary Effect
See speech act theory.

Personalization
An agent’s ability to take individual preferences and characteristics of users into account and adapt its behavior to these factors.

Proposition
A statement which can be either true or false. A closed proposition is one which contains no variables, other than those defined within the scope of a quantifier.

Protocol
A common pattern of conversations used to perform some generally useful task. The protocol is often used to facilitate a simplification of the computational machinery needed to support a given dialogue task between two agents. Throughout this document, we reserve protocol to refer to dialogue patterns between agents, and networking protocol to refer to underlying transport mechanisms such as TCP/IP.

Rational Effect (RE)
The rational effect of an action is a representation of the effect that an agent can expect to occur as a result of the action being performed. In particular, the rational effect of a communicative act is the perlocutionary effect an agent can expect the CA to have on a recipient agent.

Note that the recipient is not bound to ensure that the expected effect comes about; indeed it may be impossible for it to do so. Thus an agent may use its knowledge of the rational effect in order to plan an action, but it is not entitled to believe that the rational effect necessarily holds having performed the act.

Software Service
An instantiation of a connection to a software system.

Software System
A software entity which is not conformant to the FIPA Agent Management specification.

Speech Act
The notion of a speech act is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements, but also performs actions, e.g. a request or an assertion. In this context, a verb denoting a speech act, is called a *performative*, since saying it makes it so. See FIPA97, part 2 for more details.

Speech Act Theory
A theory of communications which is used as the basis for ACL. Speech act theory is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements,

but also performs actions. A speech act can be put in a stylised form that begins "I hereby request …" or "I hereby declare …". In this form the verb is called the performative, since saying it makes it so. Verbs that cannot be put into this form are not speech acts, for example "I hereby solve this equation" does not actually solve the equation. [Austin 62, Searle 69].

In speech act theory, communicative acts are decomposed into locutionary, illocutionary and perlocutionary acts. Locutionary acts refers to the formulation of an utterance, illocutionary refers to a categorisation of the utterance from the speakers perspective (e.g. question, command, query, etc), and perlocutionary refers to the other intended effects on the hearer. In the case of the ACL, the perlocutionary effect refers to the updating of the agent's mental attitudes.

Stationary agent
An agent that executes only upon the agent platform where it begins executing and is reliant upon it.

TCP/IP
A networking protocol used to establish connections and transmit data between hosts

User Agent
An agent which interacts with a human user.

User Dialog Management Service
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.

User ID
An identifier for a real user.

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.

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.

Wrapper Agent
An agent which provides the FIPA-WRAPPER service to an agent domain on the Internet.

4. Symbols (and abbreviated terms)

ACC:	Agent Communication Channel
ACL:	Agent Communication Language
AMS:	Agent Management System
AP:	Agent Platform
API:	Application Programming Interface
APSM:	Agent Platform Security Manager
ARB:	Agent Resource Broker
CA:	Communicative Act
CORBA:	Common Object Request Broker Architecture
DB:	Database
DCOM:	Distributed COM
DF:	Directory Facilitator
FIPA:	Foundation for Intelligent Physical Agents
FP:	Feasibility Precondition
GUID:	Global Unique Identifier
HAP:	Home Agent Platform

HTTP:	Hypertext Transmission Protocol
IDL:	Interface Definition Language
IIOP:	Internet Inter-ORB Protocol
IPMT:	Internal Platform Message Transport
IRE:	Identifying Referring Expression
OMG:	Object Management Group
ORB:	Object Request Broker
P3P:	Platform for Privacy Preferences Project
PICS:	Platform for Internet Content Selection
RE:	Rational Effect
RMI:	Remote Method Invocation, an inter-process communication method embodied in Java
SL:	Semantic Language
SMTP:	Simple Mail Transfer Protocol
SQL:	Structured Query Language
S/W:	Software System
TCP / IP:	Transmission Control Protocol / Internet Protocol
UDMA:	User Dialogue Management Agent
UDMS:	User Dialogue Management Service
UPA:	User Personalization Agent
UPS:	User Personalization Service
XML:	eXtensible Markup Language

5. Overview

Human-agent interaction is a particularly important aspect that should be taken into account to make agents practically usable. In FIPA 97 specification, human-agent interaction as not dealt with, primarily to avoid divergence by considering unlimited issues related to human-agent interaction. Now that FIPA 97 is issued and the scope is clearly delimited, the time is mature for specifying standards for human-agent interaction.

In the reference model of FIPA 98, compliant agents can interact with human users and/or other agents by manipulating user-related information. Compliant agents can offer services related to human-agent interaction; they can assist in managing the dialog between agent and human(s), or they can assist in acquiring, maintaining, and exploiting characteristic information about humans that is required for personalized interaction. That is, in the FIPA 98 reference model, there may be agents specialized on user dialog management or user personalization, although similar functionality may as well be realized within non-specialized agents.

Precisely, the present document specifies:

a User Dialog Management Service (UDMS) which wraps many types of software components for user interface allowing for ACL level of interaction between agents and human users.

a User Personalization Service (UPS) which in general allows for registration of, management of and access to user-related information needed for persaonlization. Information may be maintained and accessible as a data-base-like profile, but may also be based on observations of user behavior and constructed by a learning process.

6. Human-Agent Interaction: A Reference Model

In this section, we will put together the issues that we perceive to be important in human-agent interaction. In order to avoid *terminological confusion,* note that in "human-agent interaction" "agent" refers to a non-human entity, normally implemented in software, and that the "user" is a human (agent) who interacts with such an agent.

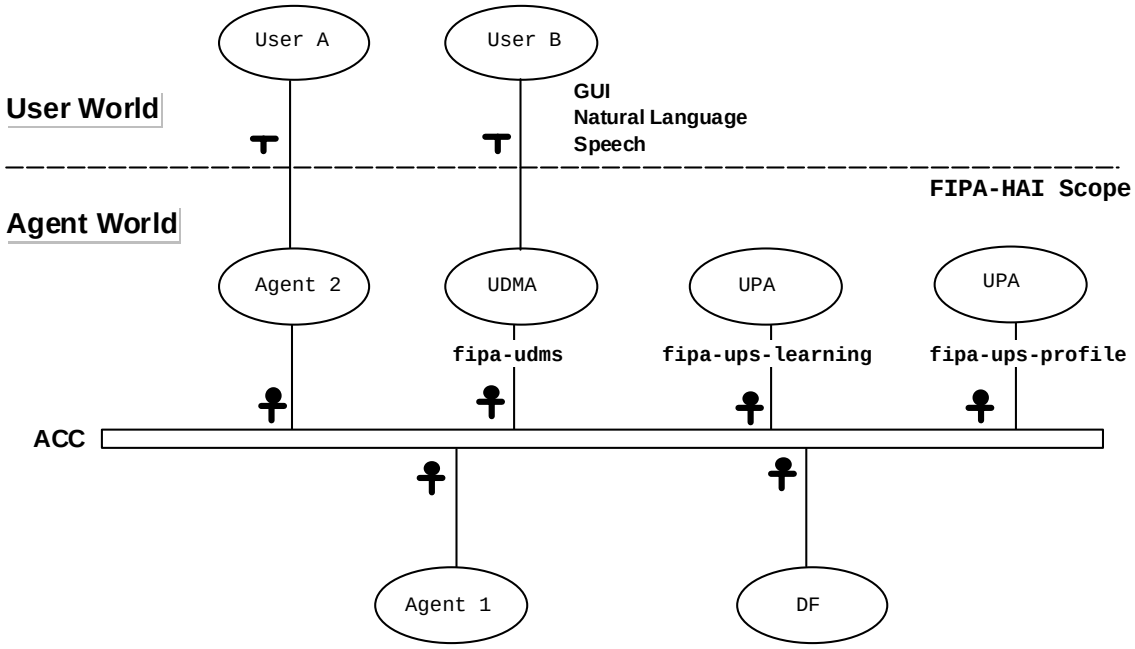


Figure 1 Human-agent interaction reference model

Figure 1illustrates 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. We will now describe these both parts in more detail.

6.1. User World

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.

6.2. Agent World

Via UDMS, interaction is possible between users and agents. 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.

In Figure 1, the user dialog management service is linked to the Agent Communication Channel (ACC), like all other entities in the agent world.

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 1, 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.

Implementers may have some good reasons for having the user model learning and the user profiling facilities separated from other functionalities. First, powerful mechanisms may be needed to acquire and represent the user models, which may easily grow too complex to be loaded into a typical agent. Moreover, these mechanisms are assumed to be generic enough to be beneficial to more than only one agent. Alternatively, implementers may also choose to directly implement profiling/learning capabilities into their agent architectures.

7. User Dialogue Management Service

7.1. Introduction

In many agent-based systems, there often occurs the need for an agent to interact with a human user. Such interaction may take place via a variety of methods, for example graphical user interface on the user's computer, speech i/o via telephone, or even simple paging. This interaction may be unidirectional (presentation to the user) or bidirectional (sending information to the user and retrieving information from the user). While many systems currently have built-in user interfaces, it is desirable to be able to take advantage of specialised I/O services offered by independent vendors. The mechanisms for offering and selecting such services can in turn best be supported by agent methodology. This section presents an overview of the actions an agent offering a User Dialogue Management Service (UDMS) may perform. Such an agent will be referred to informally in this document as a User Dialogue Management Agent (UDMA).

7.1.1. Background

FIPA 97 defines the term User Agent as an agent acting on behalf of the user. Exactly how the user agent or any other agent for that matter interacts with the user is left up to the system designer. On the other hand, mechanisms are established in FIPA 97 for agents to offer services (via a Directory Facilitator or Agent Resource Broker) and for other agents to request or negotiate about the performance of specific actions according to these services. Making use of the mechanisms offered by FIPA 97, it now becomes possible to offer, select and perform services supporting human-agent interaction.

7.1.2. Motivation

The motivation for this specification is that it allows for user interface services to be provided in an open market. The benefits are in general three fold:

Creating platform and hardware independence hence giving more portability

Offering more modalities of interaction hence providing more flexibility

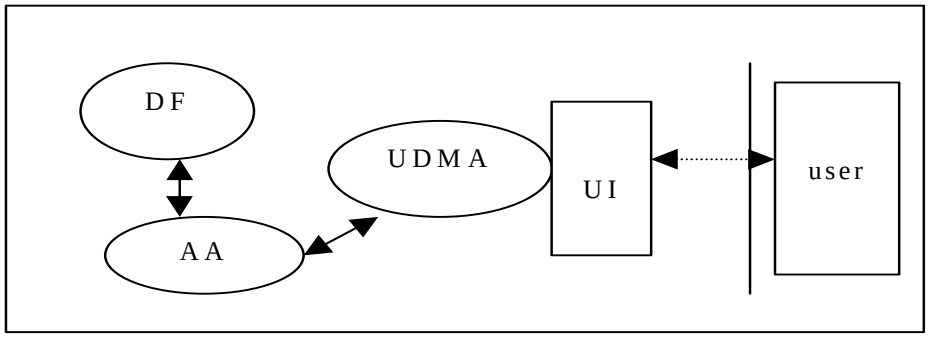
Leaving the final realisation of the visual (expression) of the user interface to the developer while indicating how the interface can be integrated with a multi-agent system.

7.2. Overview

This section presents an overview of UDMS ontology, `fipa-udms`. The conceptual designs of the UDMA indicate the abstract workings of agents wishing to interact with a UDMA for human communication. The UDMA's detailed ontology defines the basic actions necessary for an agent providing a user management dialogue service. As with any action, they may be associated with a particular quality of service and with a particular cost. The execution of these actions are the subject of negotiation between an agent requiring this service and the agent offering the service. A UDMA supplies its UDMS when registering with the directory facilitator. Clearly, a UDMA may have additional capabilities and offer additional services beyond those of user dialogue management. A UDMA need not support all of the described actions. First the conceptual design is illustrated then the lists of actions, protocols and service attributes of the ontology are given.

7.2.1. Conceptual UDMA examples

7.2.1.1. Scenario 1



DF – directory facilitator
AA – application agent
UDMA – user dialogue management agent
UI – user interface

Figure 2 Basic UDMA

In Figure 2 is the minimum conceptual UDMA and an example application agent (AA) is illustrated. We make the assumption that the application agent (AA) and the UDMA are registered with the DF. The user is registered with the DF as part of the skills capabilities of the UDMA when it registers itself with the DF (see section service description for detailed specification).

Scenario 1 illustrates the workings of a single UDMA for a single user and is agent driven in that the dialogue is initiated by an agent:

1. AA asks for an agent which can interact with the user from the DF.

2. DF returns the appropriate UDMA.

3. AA contacts the UDMA to communicate with the user and sends the content of the communication to be delivered to Peter.

In order for the UDMA to communicate the content requested for delivery by the AA the content format must be supported by the UDMA. The technical support for this is the UI part. The user interface is integral for the UDMA to supply the services (UDMS) to the application agent.

7.2.1.2. Scenario 2

Scenario 2 illustrates the conceptual design of multiple UDMAs working on behalf of one user.

In this example we are assuming there is at least one UI per UDMA and that each UI is different (e.g. supporting voice, text, video, audio etc.). All these UDMAs are associated with at least one user see Figure 3.

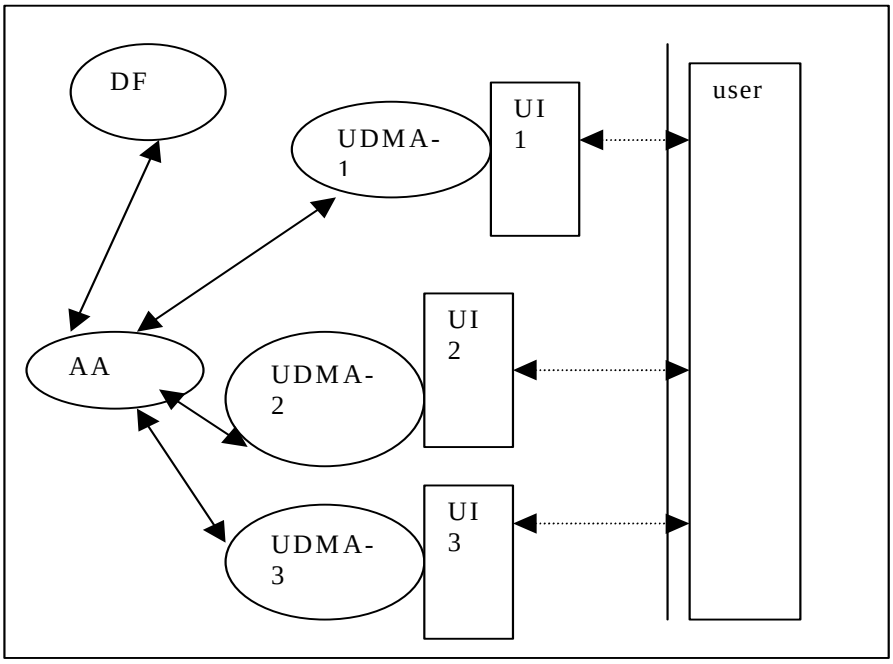


Figure 3: **Multiple UDMAs**

1. asks for an agent which can interact with the user and can handle the requested output-ontology, (e.g. voice, video) from the DF.

2. DF returns the appropriate UDMA(s).

3. If only one UDMA is returned, AA uses it. This case is the same as scenario 1.

4. If more than one UDMAs are returned, then AA needs to make a choice (the choice may be based on costs, urgency etc.). If AA want to choose a UDMA by which AA can interact the user right now, user identification actions of UDMS will be used. The choice mechanism is totally application dependent.

7.2.1.3. Scenario 3

Scenario 3 illustrates the conceptual design of broker UDMA and multiple UDMAs working on behalf of one user.

We assume here that there is a broker UDMA which can conduct other UDMAs. The conducting UDMA may be called the user's "default UDMA" or "Interface Agent" for convenience. In Figure 4, the default UDMA, UDMA-0 has no UI, however, it may directly have so-called default UI.

In this scenario, UDMAs form a domain called Domain-U by registering to DF-U. Only UDMA-0 registers external DF-A so that AA will use UDMA-0 to interact with the user.

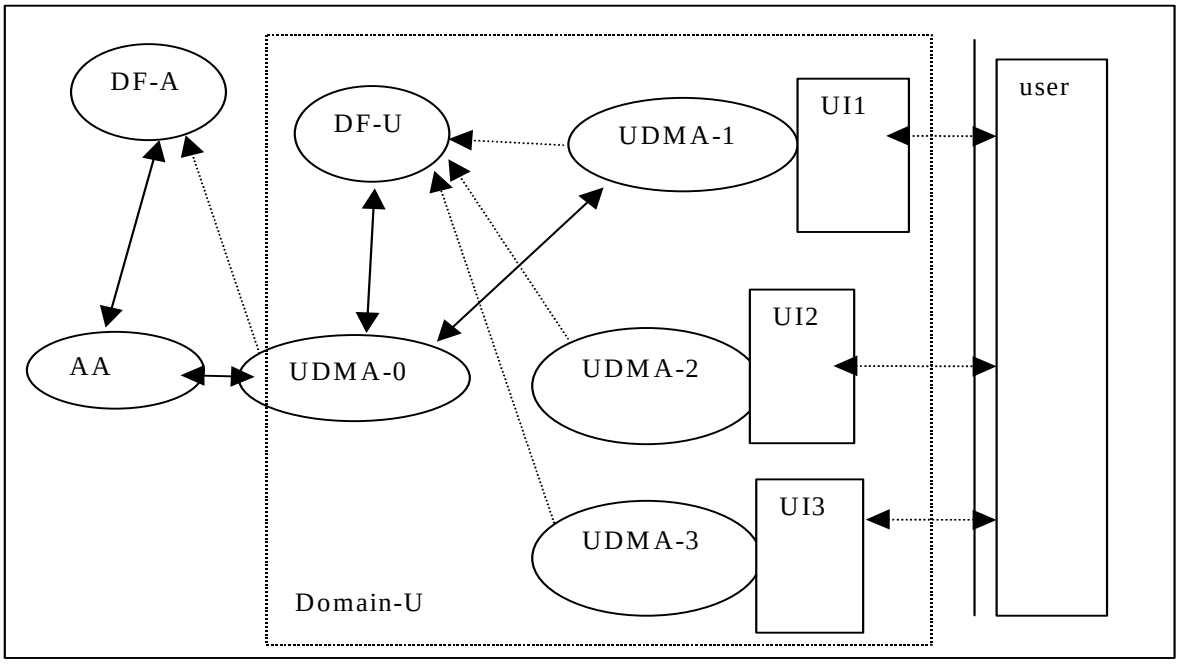


Figure 4: **Multiple UDMAs with broker**

1. AA asks for an agent which can interact with the user from DF-A.
2. DF-A returns UDMA-0.
3. AA requests user-interactions (e.g. output video) to UDMA-0.
4. UDMA-0 asks DF-U for UDMA(s) which can output the video stream to the user.
5. DF-U returns an appropriate UDMA-1.
6. UDMA-0 requests UDMA-1 to output the video stream.

UDMA-0 may switch several UDMAs or utilize them in parallel for more efficient user interactions, i.e. it may use several human-agent communication channels. User conversation control and user-conversation-id will be used to indicate the human-agent communication channel.

7.2.1.4. Scenario 4

Scenario 4 illustrates the conceptual design of multiple UDMAs for multiple users.

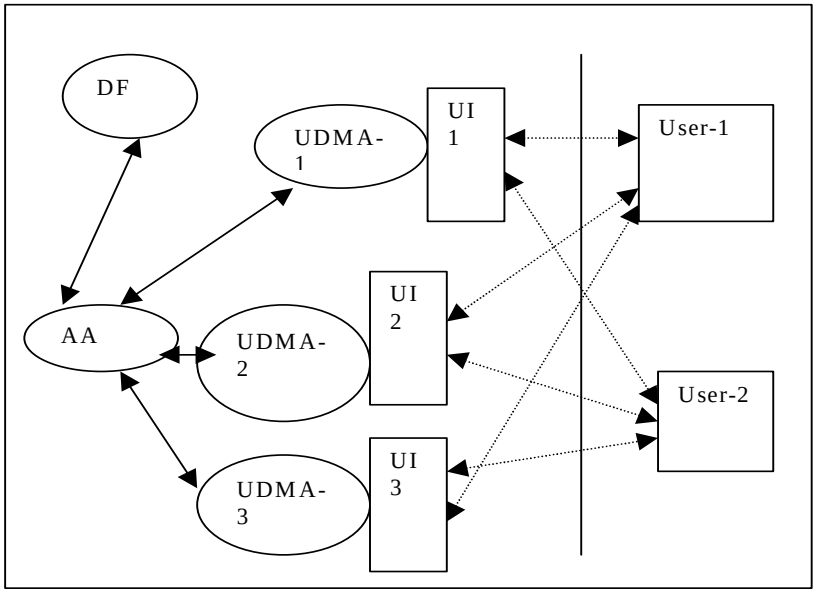


Figure 5: **Multiple UDMAs plus multiple users**

AA has the same options as stated in scenario 2 but now the UDMA needs to consider how it registers its users with DF (e.g. :user-list (user1 user2)). Also AA may have to check who is currently available to interact by user identification actions. UDMA will need to manage an internal model of how to contact each user.

7.2.1.5. UDMA supporting multiple UI

This could be applied to all previous scenarios. This reduces coordination requirements of many UDMAs to ensure the support of a variety of interfaces with the human user. However, it increases the complexity of the UDMA per se. This also effects the registration with DF as it needs to give multiple output ontology.

7.2.2. List of UDMS actions

The following table shows the actions which UDMA should support. The actions can be categorized into three types, IO actions, conversation control actions and user identification actions.

IO actions are used to interact with human users and have I/O mode to indicate whether the UDMA does input and/or output information.

Conversation control actions are used to manage conversation channel between a human user and a UDMA.

User identification actions are used to identify user(s) interacting with the UDMA synchronously (identify currently interacting user) or asynchronously (notify when the user starts interaction).

Actions	**Description**	
IO Actions		**I/O mode**
present	output information to users	o-mode, io-mode
listen	accept the input from users	i-mode, io-mode
query-user	output a question to user and get the answer	io-mode
query-if-user	output a question to user and get the answer as either true or false	io-mode
Conversation Control Actions		
start-conversation	open conversation channel between human user and UDMA	
stop-conversation	close conversation channel between human user and UDMA	
User Identification Actions		
identify-user	identify the user(s) who are currently in touch with UDMA.	
detect-user	notify when the user(s) become available to interact.	

7.2.3. List of UDMS protocols

An interaction protocol is defined to manage communication between a user and a UDMA.

Protocol	**Description**
fipa-udms-conversation	used to form a sequence of IO actions

7.2.4. UDMS service description

An agent offering a UDMS should register its service description with the DF as follows.

Attributes	**Value**
:service-type	FIPA-UDMS
:service-ontology	fipa-udms
:fixed-properties	(:modality **IO-MODE**) (:output-ontology **Supported_output_ontologies**) (:input-ontology **Supported_input_ontologies**) (:user-list **USER-IDS**)

IO-MODE = "i-mode"|"o-mode"|"io-mode"

USER-IDS = UserID+

IO-MODE denotes the agent supports only input from users (i-mode), only output to users (o-mode), or both input and output (io-mode). USER-IDS denotes the user(s) for whom the UDMA provide UDMS service.

7.2.4.1. DF Registration Example

This example explores how to register the GUI-Agent which supports both input and output for the users, John and Mary. Refer to FIPA 97 Part 1 for details.

```
(request
    :sender gui-agent@iiop://…
    :recever a-df@iiop://…
    :content
       (action a-df@iiop://…
          (register
              (:df-description
                  (:agent-name gui-agent@iiop://…)
                  (:agent-service
                      (:service-description
                          (:service-type FIPA-UDMS)
                          (:service-ontology fipa-udms)
                          (:service-name gui)
                          (:fixed-properties
                               (:modality  io-mode)
                               (:output-ontology (text image html))
                               (:input-ontology   text)
                               (:user-list (John Mary))))
                      (:interaction-protocols
                               (fipa-request fipa-udms-conversation))
                      (:ontology  fipa-agent-management)
                      (:address  iiop://…)
                      (:ownership …)
                      (:state active))))
   :language SL0
   :protocol fipa-request
   :ontology fipa-agent-management)
```

7.3. fipa-udms Ontology

This section describes detailed specifications of `fipa-udms` ontology.

7.3.1. Formal Specification

The followings are the definitions of `fipa-udms` descriptions.

7.3.1.1. fipa-udms-io-description

Parameter (content of action)	**Description**
`:user`	Denotes the ID or other reference of the user to interact with.
`:output-to-user`	Information to be presented to the user.
`:output-ontology`	Denotes the ontology in which `output-to-user` is described.
`:input-ontology`	Denotes the ontology in which the result of the input action should be described.
`:user-conversation-id`	Denotes the conversation ID if the action is performed in the sequence of the conversation.
`:constraint`	Denotes any constraints on the action
Parameter (result of action)	**Description**
`:input-from-user`	Information the user inputs. This should be returned in the format according to `input-ontology`.

7.3.1.2. fipa-udms-conversation-description

Parameter	**Description**
`:user`	Denotes the user ID to interact with.
`:user-conversation-id`	Denotes the conversation ID for `fipa-udms-conversation` protocol.
`:modality`	Denotes the modality which will be requested within the conversation. If the action requested within the conversation does not satisfy the indicated modality that action will be rejected.
`:constraint`	Denotes any constraints on the conversation

7.3.1.3. fipa-udms-identification-description

Parameter (content of action)	Description
`:candidate`	Denotes the user-id(s) to be detected.
`:trust-level-ontology`	Denotes the ontology by which `:trust-level` and `:required-trust-level` are described.
`:required-trust-level`	Denotes the minimum trust level required to authenticate.
Parameter (result of action)	**Description**
`:user`	Denotes the identified/detected user ID
`:trust-level`	Denotes how the UDMA confirms the user to interact with.

7.3.2. Attributes of fipa-udms Actions

The followings are the definitions of attributes of `fipa-udms` actions. The tables show the attributes are mandatory (**M**), optional (**O**), or not applicable (**N/A**).

7.3.2.1. IO Actions

Described by `fipa-udms-io-description`

Attribute	Action			
	`present`	`listen`	`query-user`	`query-if-user`
`:user`	M	M	M	M
`:output-ontology`	M	N/A	M	M
`:output-to-user`	M	N/A	M	M
`:input-ontology`	N/A	M	M	N/A
`:constraint`	O	O	O	O
`:user-conversation-id`	O	O	O	O

result of the action

`:input-from-user`	N/A	M	M	M

7.3.2.2. Conversation Control Actions

Described by `fipa-udms-conversation-description`.

Attribute	**Action**	
	`start-conversation`	`stop-conversation`
`:user`	M	O
`:modality`	M	O
`:constraint`	O	O
`:user-conversation-id`	M	M

7.3.2.3. User Identification Actions

Described by `fipa-udms-identification-description`.

Attribute	**Action**	
	`identify-user`	`detect-user`
`:candidate`	N/A	O
`:trust-level-ontology`	O	O
`:required-trust-level`	O	O

result of the action

`:user`	M	M
`:trust-level`	O	O

7.3.3. fipa-udms Actions

The following provides a set of actions which agents offering `fipa-udms` provide. The actions basically cover presenting and receiving information or commands to and from the user, as well as combinations thereof.

7.3.3.1. present

Supported by	fipa-udms (output)	
Description	The UDMA presents the object to the user upon request. The UDMA may convert the object into an appropriate presentation form such as speech over telephone, text on terminal display, or HTML on web browser. The agent issuing this action may indicate additional constraints such as the time constraint and/or presentation media/modality. This action has no result.	
Content	`fipa-udms-io-description`	
FIPA Protocol	`fipa-request` (see FIPA 97 Part 2), `fipa-udms-conversation`	

Example	`(request`
	`  :sender a-user-agent`
	`  :receiver phone-agent`
	`  :content`
	`  (action phone-agent`
	`  (present`
	`  (:user John)`
	`  (:output-ontology ascii-text)`
	`  (:output-to-user`
	`  "Your appointment is at 10:00")`
	`  (:constraint`
	`  (and (:before 9:00) (:language en)))`
	`  )))`
	
	possible result is "inform done"	

Refuse Reasons	`constraint-not-satisfied`	The constraint cannot be satisfied.
	`no-user`	This error occurs when the agent cannot interact with the user.
	`user-conversation-not-started`	The `user-conversation-id` is specified, but the conversation is not started.
	`output-ontology-not-supported`	The output ontology is not supported by the UDMA.
	`output-content-mismatched`	The output content is not of the type specified by the `output-ontology`.
Failure Reasons		

7.3.3.2. listen

<table>
<tr><td>Supported by</td><td colspan="2">fipa-udms (input)</td></tr>
<tr><td>Description</td><td colspan="2">The UDMS accepts input from the user. The result of the action is the input, which is generally returned to the agent requesting this service (via a fipa-request protocol).</td></tr>
<tr><td>Content</td><td colspan="2"><code>fipa-udms-io-description</code></td></tr>
<tr><td>FIPA Protocol</td><td colspan="2"><code>fipa-request</code> (see FIPA 97 Part 2), <code>fipa-udms-conversation</code></td></tr>
<tr><td>Example</td><td colspan="2">Sending the message
<pre>(request
      :sender a-user-agent
      :receiver phone-agent
      :reply-with abc123
      :content
      (action phone-agent
       (listen
         (:user John)
         (:input-ontology ascii-text)
         (:constraint (:before 9:00))
         )))</pre>may result in
<pre>(inform
      :sender phone-agent
      :receiver a-user-agent
      :in-reply-to abc123
      :content
        (result
          (action phone-agent (listen …))
          (:input-from-user "I can't attend")
        ))</pre></td></tr>
<tr><td rowspan="4">Refuse Reasons</td><td><code>constraint-not-satisfied</code></td><td>The constraint cannot be satisfied.</td></tr>
<tr><td><code>no-user</code></td><td>This error occurs when the agent cannot interact with the user.</td></tr>
<tr><td><code>user-conversation-not-started</code></td><td>The user-conversation-id is specified, but the conversation is not started.</td></tr>
<tr><td><code>input-ontology-not-supported</code></td><td>The input ontology is not supported by the UDMA.</td></tr>
</table>

7.3.3.3. query-user

<table>
<tr><td>Supported by</td><td colspan="2">fipa-udms (I/O)</td></tr>
<tr><td>Description</td><td colspan="2">The UDMS asks the user a question.

Although query-user can be represented by using present and listen with appropriate synchronization constraints, it is easier to represent this as a single action.</td></tr>
<tr><td>Content</td><td colspan="2">fipa-udms-io-description</td></tr>
<tr><td>FIPA Protocol</td><td colspan="2">fipa-request (see FIPA 97 Part 2), fipa-udms-conversation</td></tr>
<tr><td>Example</td><td colspan="2">(request
:sender a-user-agent
:receiver phone-agent
:reply-with abc456
:content
(action phone-agent
(query-user
(:user John)
(:output-ontology audio)
(:output-to-user
encoded audio format saying "Can you attend the 10AM meeting?")
(:input-ontology ascii-text)
(:constraint (:before 9:00))
)))

may result in

(inform
:sender phone-agent
:receiver a-user-agent
:in-reply-to abc456
:content
(result (action phone-agent (query-user …))
(:input-from-user "I can't attend")
))</td></tr>
<tr><td rowspan="6">Refuse Reasons</td><td>constraint-not-satisfied</td><td>The constraint cannot be satisfied.</td></tr>
<tr><td>no-user</td><td>This error occurs when the agent cannot interact with the user.</td></tr>
<tr><td>user-conversation-not-started</td><td>The user-conversation-id is specified, but the conversation is not started.</td></tr>
<tr><td>output-ontology-not-supported</td><td>The output ontology is not supported by the UDMA.</td></tr>
<tr><td>output-content-mismatched</td><td>The output content is not of the type specified by the output-ontology.</td></tr>
<tr><td>input-ontology-not-supported</td><td>The input ontology is not supported by the UDMA.</td></tr>
</table>

7.3.3.4. query-if-user

<table>
<tr><td>Supported by</td><td colspan="2">fipa-udms</td></tr>
<tr><td>Description</td><td colspan="2">The UDMS asks the user a yes/no question.</td></tr>
<tr><td>Content</td><td colspan="2"><code>fipa-udms-io-description</code></td></tr>
<tr><td>FIPA Protocol</td><td colspan="2"><code>fipa-request</code> (see FIPA 97 Part 2), <code>fipa-udms-conversation</code></td></tr>
<tr><td>Example</td><td colspan="2">Sending the message

<code>(request
:sender a-user-agent
:receiver phone-agent
:reply-with abc789
:content
(action phone-agent
(query-if-user
(:user John)
(:output-ontology audio)
(:output-to-user</code>
encoded audio format saying "Can you attend the 10AM meeting?")
<code>(:constraint (:before 9:00))
)))</code>

may result in

<code>(inform
:sender phone-agent
:receiver a-user-agent
:in-reply-to abc789
:content
(result (action phone-agent (query-if-user …))
(:input-from-user false)
))</code></td></tr>
<tr><td rowspan="5">Refuse Reasons</td><td><code>constraint-not-satisfied</code></td><td>The constraint cannot be satisfied.</td></tr>
<tr><td><code>no-user</code></td><td>This error occurs when the agent cannot interact with the user.</td></tr>
<tr><td><code>user-conversation-not-started</code></td><td>The user-conversation-id is specified, but the conversation is not started.</td></tr>
<tr><td><code>output-ontology-not-supported</code></td><td>The output ontology is not supported by the UDMA.</td></tr>
<tr><td><code>output-content-mismatched</code></td><td>The output content is not of the type specified by the output-ontology.</td></tr>
</table>

7.3.3.5. start-conversation

Supported by	fipa-udms	
Description	The agent interacting with the user by using UDMS may want to control the conversation. This action is used to start the `fipa-udms-conversation` protocol. The UDMS agent executing action may open a dialog window (in case of GUI) or call the user's phone (in case of telephone agent). Action has no result. The agent requesting this action must specify unique `user-conversation-id`, and it has to be specified in actions involved in the protocol. The `user-conversation-id` is associated with each action, while conversation-id defined in FIPA 97 Part 2 is associated with each message to indicate agent-agent conversation.	
Content	`fipa-udms-conversation-description`	
FIPA Protocol	`fipa-udms-conversation`	

Example	`(request`
	`:sender a-user-agent`
	`:receiver phone-agent`
	`:content`
	`(action phone-agent`
	`(start-conversation`
	`(:user John)`
	`(:user-conversation-id conv111)`
	`(:modality io-mode)`
	`(:constraint (:before 9:00))`
	`)))`	

Refuse Reasons	`no-user`	This error occurs when the agent cannot interact with the user.
	`constraint-not-satisfied`	The constraint cannot be satisfied.
	`modality-not-supported`	The modality cannot be supported.

7.3.3.6. stop-conversation

Supported by	fipa-udms	
Description	The agent interacting with the user by using UDMS may want to control the conversation. This action is used to stop the conversation. The UDMS agent executing action may close the dialog window (in case of GUI) or hang up the phone (in case of telephone agent). Action has no result.	
Content	`fipa-udms-conversation-description`	
FIPA Protocol	`fipa-udms-conversation`	

Example	`(request`
	`:sender a-user-agent`
	`:receiver phone-agent`
	`:content`
	`(action phone-agent`
	`(stop-conversation`
	`(:user-conversation-id conv111)`
	`)))`	

Refuse Reasons	`user-conversation-not-started`	The user-conversation-id is specified, but the conversation is not started.

7.3.3.7. identify-user

Supported by	fipa-udms	
Description	The UDMS identifies the user(s) who are currently in touch via this UDMS. This action can be used to authenticate the user with a certain trust level.	
Content	`fipa-udms-identification-description`	
FIPA Protocol	`fipa-request`	

Example	`(request`
	`  :sender  an-agent`
	`  :receiver  gui-agent`
	`  :reply-with  abc123`
	`  :content  (action  gui-agent`
	`  (identify-user`
	`  (:trust-level-ontology`
	`  e-commerce-authentication)`
	`  (:required-trust-level  4digit-pin)`
	`)))`
	
	may result in
	
	`(inform`
	`  :sender  gui-agent`
	`  :receiver  an-agent`
	`  :in-reply-to  abc123`
	`  :content`
	`  (result`
	`  (action gui-agent (identify-user …))`
	`  ((:user  John)`
	`  (:trust-level-ontology`
	`  e-commerce-authentication)`
	`  (:trust-level`
	`  (and  4digit-pin  license-number)))`
	`  ))`	

Refuse Reasons	`no-user`	The UDMS is not currently interacting with the user.
	`no-trust-user`	The UDMS is interacting with the user but the trust level is not as high as required.

7.3.3.8. detect-user

Supported by	fipa-udms	
Description	The UDMS notifies the requesting agent when the user(s) are available to interact via this UDMS. If the `candidate` is specified, the UDMA replies only when any of the candidates is available.	
Content	`fipa-udms-identification-description`	
FIPA Protocol	`fipa-request`	

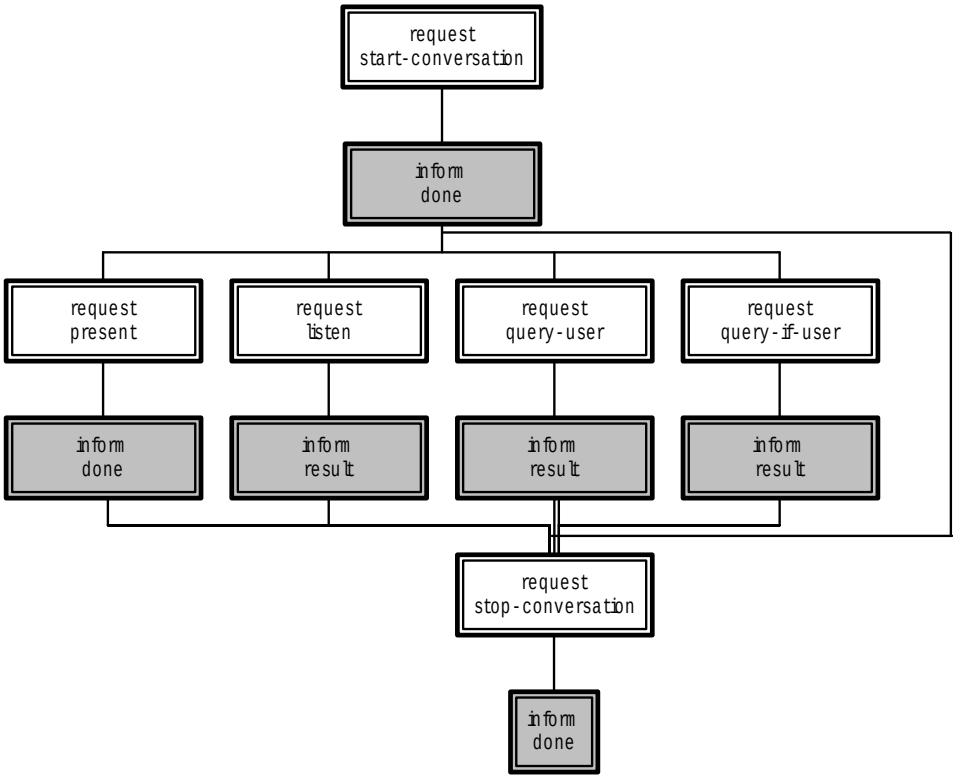
Example	`(request`
	`  :sender  an-agent`
	`  :receiver  kiosk-agent`
	`  :reply-with  abc123`
	`  :content`
	`  (action kiosk-agent`
	`  (detect-user  (:candidate  (John Tom Mary)))`
	`))`
	
	may result in the following when John comes to login the kiosk terminal
	
	`(inform`
	`  :sender  kiosk-agent`
	`  :receiver  an-agent`
	`  :in-reply-to  abc123`
	`  :content`
	`  (result`
	`  (action kiosk-agent (detect-user …))`
	`  (:user John)`
	`  ))`	

Refuse Reasons		

7.3.4. fipa-udms Interaction Protocols

Use `fipa-request` and/or `fipa-contract-net` to support negotiation and handling of "one time" user interaction. If a sequence of interactions with user (a conversation) is foreseen, the following interaction protocol may be used:

`fipa-udms-conversation`



7.3.4.1. Example

Suppose John's Personal Scheduler Agent (PSA) gets two requests to arrange his schedule from Tom and Mary.

1. PSA searches UDMA interacting with John through DF in order to confirm his schedule arrangements. Suppose a GUI based Agent (GUIA) is selected.

2. PSA sends start-conversation (`:user-conversation-id 1`) and start-conversation (`:user-conversation-id 2`).

3. GUIA opens two window (say Window A and Window B).

4. PSA sends present, query-user, query-if-user, and/or listen actions with (`:user-conversation-id 1`) to confirm his schedule requested by Tom. PSA does with (`:user-conversation-id 2`) for Mary.

5. The GUIA shows messages of (`:user-conversation-id 1`) on Window A and messages of (`:user-conversation-id 2`) on Window B so that John can see and choose to whom he is making arrangements of his schedule.

6. When PSA sends stop-conversation with (`:user-conversation-id 1 / 2`), the GUIA closes Window A / B to indicate the conversation ends.

Note: PSA can interact with John without `fipa-udms-conversation`. In the case, however, PSA may have to serialize the requests from Tom and Mary, for John not to confuse to which request he is interacting with otherwise the agent internally must manage multiple dialogues .

7.4. Working scenarios

The following scenarios provide some pragmatic examples of how to use the actions with the UDMA when there is either one UDMA or more. This is just for illustrative purposes (informative).

7.4.1. Scenario 1

Scenario 1 (see Section 7.2.1.1) illustrates the workings of a single UDMA with a single user. The action is defined in the content and the communicative act is given, sender and receiver are source-agent and destination agent.

Agent Source	**S:D**	**Agent Dest**	**Action**
AA	1:1	DF	(request … :content (action DF (search (:df-description (:service-description (:service-type FIPA-UDMS) (:service-ontology fipa-udms) (:fixed-properties (:modality io-mode :output-ontology video) :user-list Madison))) … :reply-with AA001)
DF	1:1	AA	(inform … :content (result (action df (search …)) (:agent-name UDMA)) :in-reply-to AA001)
AA	1:1	UDMA	(request … :content (action UDMA (present (:user Madison) (:output-ontology video) (:output-to-user Wild-animals-clip) (:constraint (and (:before <time>) (:language <language>))))) :reply-with AA002)
UDMA	1:1	AA	(inform … :content (done present) :in-reply-to AA002)

7.4.2. Scenario 3

Scenario 3 (see Section 7.2.1.3) using a UDMA assuming that there is always at least one UDMA registered with the DF which knows about a user Madison but does not have direct user interface of the requested media (`output-ontology`). The UDMA has to find another UDMS service to support the AA communication with the user, Madison.

Agent Source	**S:D**	**Agent Dest**	**Action**
AA	1:1	DF-A	(request … :content (action DF-A (search (:df-description (:service-description (:service-type FIPA-UDMS) (:fixed-properties (:modality io-mode :user-list Madison)))))) … :reply-with AA001)
DF-A	1:1	AA	(inform … :content (result (action …) (:agent-name UDMA-0)) :in-reply-to AA001)
AA	1:1	UDMA-0	(request …:content (action UDMA-0 (present (:user Madison) (:output-ontology video) (:output-to-user Wild-animals-clip) (:constraint (and (:before <time>) (:language <language>))))) :reply-with AA002)
UDMA-0	1:1	DF-U	(request … :content (action DF-U (search (:df-description (:service-description (:service-type FIPA-UDMS) (:fixed-properties (:modality io-mode :output-ontology video :user-list Madison)))))) … :reply-with UDMA001)
DF-U		UDMA-0	(inform … :content (result (action …) (:agent-name UDMA-1)) :in-reply-to UDMA001)
UDMA-0		UDMA-1	(request …:content (action UDMA-1 (present (:user Madison) (:output-ontology Video) (:output-to-user Wild-animals-clip) (:constraint (and (:before <time>) (:language <language>)))))

			:reply-with UDMA002)
UDMA-1		UDMA-0	(inform … :content (done present) :in-reply-to UDMA002)
UDMA-0		AA	(inform … :content (done present) :in-reply-to AA002)

8. User Personalization Service

8.1. Motivation and Introduction

As far as human-agent interaction is concerned, the intelligence of an agent will be mainly determined by its ability of *personalization*, i.e., to take individual preferences and characteristics of users into account and adapt its behavior to these factors. Personalization has perhaps first been an issue in natural-language dialog systems, where both interpretation of human utterances as well as machine generation of appropriate dialog contributions can be improved by taking especially knowledge and goals of human dialog partners into account. Later, the ability to personalize has been regarded as the predominant characteristic of personal assistants or interface agents, which appeared together with the research area of intelligent user interfaces.

While the driving force of personalization research has traditionally been the goal to improve usability, supportiveness, and effectiveness of interactive software systems, interest in one-to-one marketing as a central means of electronic commerce has recently increased the demand for personalization capabilities dramatically. The need for personalization within agent environments was demonstrated by the FIPA97 applications, all of which involved some kind of personalization component. For example, in FIPA97 Part 6, a “user profile agent” was employed as a distinguished entity for maintaining personalization information.

Most obviously, information about the user upon which personalized behavior can be based must be available. Such information must be acquired in some way and, if it is of at least mid-term interest, it must be kept permanently within an information store, which we will refer to as user model. We envision two approaches to deal with user models: First, explicitly formulated information items about a user can be maintained like in a data base (or a knowledge base, if sophisticated representation and inference mechanisms shall be involved). Data-base-like user models have often been called user profiles. Typically, user profile contents are explicitly entered into the profile.

However, there are many scenarios where information about the user is not explicitly available but must be acquired from observations of user behavior. In these cases, complex learning mechanisms may be needed, which result in association-like information about a user. Access to learning-based user models differs signifcantly from access to profile-like user models.

Hence, two UPS subtypes will be introduced, one with a simpler interface for dealing with user profiles, the other with a more complex interface for dealing with learning tasks.

Both functionalities will often be private to user agents that are to show personalized behavior. However, it may often be beneficial to delegate these personalization tasks to a third party. This has several positive consequences:

User agents can be released from performing possibly complex user model learning, representation, maintenance, and reasoning tasks.

User models can be based on the interactions between users and more than one agent, which may lead to richer user models.

User models can be available to more than one agent, which avoids repeated acquisition of the same information and hence may decrease obtrusiveness of the human-agent interface.

This section provides a unified view of personalization within the FIPA agent framework and specifies the requirements that any agent offering personalization facilities to others is to meet. Note, however, that the definition of a distinguished service does not preclude the implementation of agent-specific personalization facilities.

A User Personalization Service (UPS) is specified that may be offered by agents. Such a service may include either standard profiling or learning capabilities needed for personalization (an agent is free to establish more than one UPS, if both capabilities shall be provided). An agent that offers a user personalization service will be referred to informally in this document as User Personalization Agent (UPA). This section normatively describes the actions that such an agent may perform and specifies all details necessary to make use of its service. It also defines a user model description object that is needed to uniquely identify each user model maintained by a UPS. Such a user model description mainly consists of a user ID and a user model type; the type determines possible contents of and ways to access a user model. Moreover, there may be multiple instances of a user model with a certain type, so that different situational contexts of the user can be taken into account.

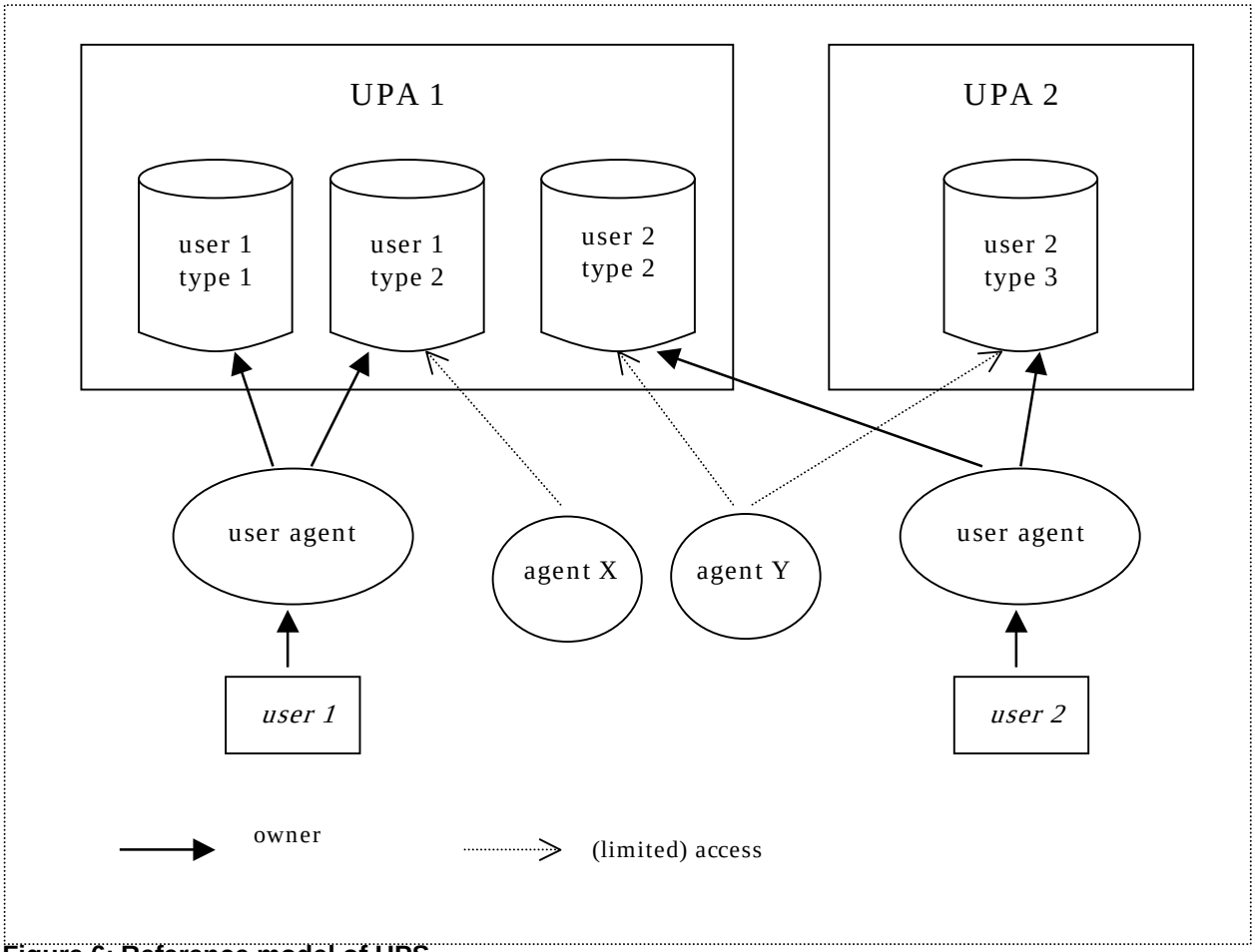


Figure 6: Reference model of UPS

Every user model that is maintained by a UPS is owned by the agent that requested its creation. Access to the user model by other agents is governed by access conditions which only the owning agent may set and modify. The details of access conditions are described in a later subsection.

Figure 6 presents a UPS reference model. It shows two UPAs maintaining several user models identified by a user model description. User agents are shown that own these models since they requested their creation. They have full access, while other agents that do not own these models and hence may have only limited access are also depicted.

8.2. Formal Overview

In this section, we specify the service attributes of a UPS (as required as parts of a `FIPA-Service-Desc`) and briefly list both the actions and the protocols that a UPS supports. In the following subsections, these actions and protocols will be described in detail.

Service Attributes of a UPS:

Attribute	Value
:service-type	FIPA-UPS
:service-ontology	fipa-ups | fipa-ups-profile | fipa-ups-learning
:fixed-properties	a list of user model types supported

A UPS uses one of three possible ontologies. With each ontology, a set of actions and protocols that are supported by the UPS is associated. They are described in the following sections. Typically, the ontology will be one of fipa-ups-profile or fipa-ups-learning, since fipa-ups does not allow access to the user model. The fipa-ups ontology provides basic user modeling abilities, which are inherited by the other two ontologies, which provide different means for user model access.

Using the :fixed-properties service attribute, a list of supported user model types is given. A user model type determines structure and possibe contents of user models (see Section 8.3.1).

```
(request
      :sender up-agent@iiop://fipa.org:50/acc
      :receiver a-df@iiop://fipa.org:50/acc
      :content
            (action a-df@iiop://fipa.org:50/acc
                  (register
                    (:df-description
                        (:agent-name up-agent@iiop://fipa.org:50/acc)
                        (:agent-services
                          (:service-description
                              (:service-type FIPA-UPS)
                              (:service-name ups-1)
                              (:service-ontology fipa-ups-profile)
                              (:fixed-properties
                                (:um-types P3P travel-prefs vCard))))
                        (:interaction-protocols
                              (fipa-query
                               fipa-request-when
                               fipa-request
                               fipa-iterated-contract-net))
                        (:ontology  fipa-agent-management)
                        (:address   iiop://fipa.org/acc)
                        (:ownership fipa.org)
                        (:state     active))))
      :language   SL
      :protocol   fipa-request
      :ontology   fipa-agent-management)
```

Figure 7: Registration of a UPS

Figure 7 illustrates how a UPS that supports the ontology fipa-ups-profile registers with a directory facilitator using “ups-1” as service name. It supports the user model types “P3P”, “travel-prefs”, and “vCard”.

As part of this formal overview, the following tables list the protocols that any agent offering the FIPA-UPS service (i.e., any UPA) supports and the actions that are part of the basic fipa-ups ontology (and hence also supported by any UPA). The input arguments will be explained in the following subsections. The actions themselves are defined in Section 8.3.3. Further actions that are part of the more specific ontologies will be described in the respective sections.

Protocols supported by any UPA:

Protocol Name
FIPA-Query
FIPA-Request-When
FIPA-Request
FIPA-Iterated-Contract-Net

Actions in the fipa-ups ontology, supported by any UPA

Action	Input arguments
create-user-model	user model description
delete-user-model	user model description
register	user model description, access condition
set-privacy-control	user model description, access condition

8.3. fipa-ups Ontology

8.3.1. User model description

FIPA does not make its own assumptions about structure and possible contents of user models. A FIPA-compliant user model is identified by the user it models and by a user model type, which determines structure and possible contents of the user model. For a user, several models may exist, perhaps distributed among several UPAs. Within one UPA, however, there must be only one model of a given type for one user. That is, in communications with a UPA, the relevant user model is uniquely specified by a user identification (user ID) and a user model type.

User ID and user model type identify a user model as a whole. Typically, agents will want to access (read or write) only a part of a user model. What a user model part is, is determined by the type of the user model. Access pointers will provided as the generic means to access user model parts.

There may even be more parameters that are useful when accessing a user model. E.g., an agent querying a user model may want to ask the UPA to send the results in a specific format.

This section explains all parameters needed for user model access in more detail and formally defined an object for user model description, which collects these parameters.

8.3.1.1. Parameters for user model description and access

User ID

In the following, a mechanism for generating a unique user ID is assumed. Typically, this will be a kind of login procedure to the agent system. Uniqueness means that it can be guaranteed that the ID refers to only one user. A bijective mapping is not assumed, i.e., a real human user may be referred to by more than one user ID. A UPA will not try to detect if different user IDs refer to the same user; different IDs lead to different user models.

User Model Type

In general, there is no reason to assume that there will be only one user model for a user. There may be many sources of information about a user that may refer to one or more different situational contexts, as, for example, basic personal information (e.g., address, phone number), preferences in TV program selections, interaction modalities (e.g., GUI, speech, etc) with computer systems or educational status of the user. In order to preserve the necessary level of openness and genericity, FIPA does not adopt any specific model, but allows to identify the type of a user

model. A user model type implies structure and possible contents of a user model, as well as means to access parts of a user model (see next subsection). Complex mechanisms to define user model types can be imagined. However, for the purpose of this specification, we assume that a user model types are referred to by unique type IDs. Application developers must take care of the sound use of such user model type IDs.

An example is the type “P3P”. This ID may be used to refer to a user model, the structure and elements of which obeys the specifications of the P3P group of the W3C. A P3P user model consists of data elements (attribute-value pairs) that can be grouped into data sets. Data sets can again be grouped, so that a hierarchical structure is imposed. The P3P specifications require certain data sets to be present, e.g. a set “User” containing basic demographic information like birth date and gender as well as basic contact information.

User Model Instance

Users may have different roles or be part of different contexts. This enforces the need for several instances of a user model, which have the same type (i.e., structure, possible contents), but have different actual contents. For example, the travel preferences of a user may be modeled once for business trips and once for vacation traveling. The user model instance parameter can be optionally used in most UPS actions to identify a specific user model instance.

User Model Component

When retrieving from or adding to a user model, in most cases only a part or component of the user model is concerned. For example, in a P3P user model, the gender of the user may be retrieved from the model, or perhaps the whole “User” data set.

A parameter is provided that allows to identify the user model component to be accessed. The form of such an identifier is typically specific to the user model type. E.g., for P3P models, a “dot notation” must be used that points into the model hierarchy by concatenating hierarchy notes, separated with a dot. In the above example, “User.gender” would be the appropriate user model component identifier, while “User.” would refer to the whole data set.

In other user model types, user model components may look completely different. For example, the user modeling shell system BGP-MS[1] employs a modal logic language to express beliefs about the user’s mental attitudes (like goals, intentions, beliefs, etc.). An example is the expression `(B S (W U (arrange_meeting (Jack Jill))))`, which represents a belief (B) of the system (S) that the user (U) wants (W) a meeting with a group of other participants to be arranged. In BGP-MS, the user model is a modal logic knowledge base. Each well-formed expression in the modal logic language is a component of the user model, even if it is not explicitly contained in the knowledge base; an inference engine can be employed to determine the value of the component.

User Model Language

This parameter can be used to specify the language that is employed to transfer user model contents. For example, a query to a user model with type vCard and language set to XML might result in the following (no access pointer is given, so that a complete vCard model is returned):

```
<vcard>
      <fn:>  Peter Baumann </fn:>
      <org:> Tangerine Dream Music Co. </org:>
      <adr:> F. Wellesplein 1;;;Antwerpen;;B-0000;Belgium </adr:>
      <email;>
            <internet> baumann@tdm.com </internet>
      </email:>
      <title:> Advanced System Development Engineer </title:>
      <tel;>
            <work:> +32 3 000 11 11</work>
            <fax:>  +32 3 000 22 22</fax>
      </tel>
<x-mozilla-cpt:>  ;0 </x-mozilla-cpt:>
<x-mozilla-html:> TRUE </x-mozilla-html:>
<version:>        2.1  </version:>
</vcard>
```

This parameter may also be used in the creation of a user model. In this case it sets a default language for transporting user model contents.

[1] Kobsa, A. & Pohl, W.: The User Modeling Shell System BGP-MS. User Modeling and User-Adapted Interaction 4(2), .59-106.

User Model Ontology

When creating a user model, an ontology may be given that tells the UPS how to interpret user model contents. User model ontologies may enable content transfer between user models.

8.3.1.2. Formal specification

This section defines the object `fipa-ups-um-description`, which summarizes parameters needed by most UPS actions. Like all other object definitions in this document, this definition is extensible, in that additional parameters can be defined and used by agent developers.

fipa-ups-um-description

Parameter	Description
`:user-id`	Denotes a unique user identifier associated with a user model (e.g., Joe, ID887, etc).
`:um-type`	Identifies type of the user model described (e.g., P3P, Vcard, PICS, NPS, etc).
`:um-instance`	Denotes a specific instance of a user model, the type of which is given by `:um-type`.
`:um-component`	Denotes a component of user information stored in the user model (e.g., “tel.office” denotes office telephone number in a user model).
`:um-language`	Determines syntactic format to be used for transporting user model contents (e.g., XML, SL, etc) .
`:um-ontology`	Denotes domain information about the user model type (e.g., business, shopping, etc) .

The following table summarizes the use of these parameters by the global UPS actions that are defined in Section 8.3.3 (M = mandatory, O = optional, N/A = not applicable):

Action / **Parameter**	`create-user-model`	`delete-user-model`	`register`	`set-access-control`
`:user-ID`	M	M	M	M
`:um-type`	M	M	M	M
`:um-instance`	O	O	O	O
`:um-component`	N/A	N/A	O	O
`:um-language`	O	O	O	O
`:um-ontology`	O	O	O	O

8.3.1.3. Registering user models with the DF

A UPA registers with the DF the user models that it maintains by using user model descriptions. For this purpose, the `:negotiable-properties` slot of the FIPA-DF-description agent description object. This allows user model

consumers to easily find the right UPA. The user

Assume for example that, upon request of an agent, a UPS creates a user model to maintain the business travel preferences of user Joe. As a consequence, it sends the following message to its directory facilitator.

```
(request
  :sender up-agent@iiop://fipa.org:50/acc
  :receiver a-df@iiop://fipa.org:50/acc
  :content
    (action a-df@iiop://fipa.org:50/acc
            (modify
              (:df-description
                (:agent-services
                  (:service-description
                    (:negotiable-properties
                      (:user-models
                        (:user-ID joe
                         :um-type travel-prefs
                         :um-instance business-travel)
                        <descriptions of all other available user models>
                        )))))))
  :language SL
  :protocol fipa-request
  :ontology fipa-agent-management)
```

8.3.2. Access Control

User models, whether simple and profile-like or complex and learned, contain sensitive information, the use of which needs to be controlled. There may be different motivations for access control: On the one hand, users may want to regulate the use of private data they have actively provided just to make sure who knows what about them. On the other hand, commerce sites that have acquired user-related data (e.g., by using a learning UPS) may want to protect this precious source of information.

In this section, we set up a simple access control model for a FIPA-UPS. This model allows owners of user models to establish access permissions (read, write, read-write) relative to properties of potential user model consumers, as they are available via DFs.

This specification does not deal with security issues. Relevant security issues are

source authentication of the user model consumer: This is a prerequisite if one wants to be sure about sending user model contents to permitted parties only.

message confidentiality: It may be desirable to send user model contents to user model consumers confidentially, so that third parties cannot get hold of the data

message integrity: User model contents should not get corrupted on their way to the user model consumer, since this would render adaptation unsound or even impossible.

Mechanisms to deal with these issues are provided in FIPA 98, part 10 (Agent Security Management). They can be used in the communication between a UPS and user model consumers, but their use is not required. However, a UPS may simply reject user model access requests the authenticity of which is not guaranteed.

8.3.2.1. Access Permissions

First, a set of constants is needed, each of which denotes an access permission. Access permissions will be associated with user model parts. What can be a user model part depends on the user model type, particularly its user model access syntax, which may allow to identify whole sections of a user model or not. The following table defines the syntactic category fipa-ups-access-permission and its three alternative terminal values.

fipa-ups-access-permission

`fipa-ups-read`	permission to get the associated user model part transferred

	transferred
`fipa-ups-write`	permission to modify the associated user model part
`fipa-ups-readwrite`	permission to both modify and get transferred the associated user model part

8.3.2.2. Partial Agent Description

An access permission is relative to a FIPA-DF-description, as it is defined in FIPA 97, Part. 1. An example of such an agent description is the following:

```
(:df-description
  (:agent-name an-agent@iiop://fipa.org:50/acc)
  (:agent-services
    (:service-description
      (:service-type video-on-demand)
      (:service-ontology itut-vod)
      (:service-name ntt-vod)))
  (:interaction-protocols (fipa-request))
  (:ontology fipa-agent-management)
  (:address iiop://fipa.org/acc)
  (:ownership fipa.org)
  (:state active))
```

User model access permissions can be given relative to *partial agent descriptions*, which specify values for a subset of the attributes of a FIPA-DF-description. The missing attributes can be considered as “wild cards”; they are not checked to see if an agent matches a given partial description. An example of a partial description (which, nevertheless, is a legal instance of FIPA-DF-description) is

```
(:df-description (:ownership fipa.org))
```

It matches all agents the owner of which is identified by “fipa.org”.

8.3.2.3. Access Condition

With access permissions and partial agent descriptions, we now have the two constituents of access conditions at hand. That is, an access condition specifies a certain access permission for all agents matching a certain partial agent description.

We define a fipa-ups-access-condition object that summarizes these two components.

fipa-ups-access-condition

Parameter	**Description**
`:access-permission`	a fipa-ups-access-permission
`:accessor-description`	a FIPA-DF-description (see FIPA 97, part 1) that specifies the set of agents the above permission is assigned to

An example is:

```
(:access-permission fipa-ups-read
 :accessor-description (:df-description (:ownership fipa.org)))
```

This condition gives read permission to all agents with ownership equal to “fipa.org”.

As already mentioned above, access conditions may be assigned to whole user models or user model components.

See Section 8.3.1.1 for how user model components are identified, and see the specification of the UPS action `set-access-control` (Section 8.3.3.4) for how access conditions can be assigned to user model parts.

8.3.3. UPS Actions

8.3.3.1. create-user-model

Supported by	fipa-ups	
Description	The UPS sets up a user model according to the given description. The requesting agent is registered as owner of the created model. If there is already a user model for this description, the action is refused. The requested action fails if the UPA does not have sufficient resources.	
Content	`fipa-ups-um-description`	
FIPA Protocol	`fipa-request`	

Example	`(request :sender user-agent`
	`:receiver UPA`
	`:content (action UPA`
	`(create-user-model`
	`(:user-id joe`
	`:um-type vCARD`
	`:um-language XML`
	`:um-ontology business)))`
	`:language SL`
	`:protocol fipa-request`
	`:reply-with upa-req3`
	`)`	

Refuse Reasons	`already-created`	There already is a user model maintained for the given ID and type.
Failure Reasons	`out-of-resources`	The UPA cannot create the specified user model due to resource constraints.

8.3.3.2. delete-user-model

Supported by	fipa-ups	
Description	The UPS deletes the specified user model. If the requesting agent is not owner of the model, the request is refused. The request fails if the specified model does not exist.	
Content	`fipa-ups-um-description`	
FIPA Protocol	`fipa-request`	

Example	`(request :sender user-agent`
	`:receiver UPA`
	`:content (action UPA`
	`(delete-user-model`
	`(:user-id joe`
	`:um-type vCARD)))`
	`:language SL`
	`:protocol fipa-request`
	`:reply-with ume-req3`
	`)`	

Refuse Reasons	`not-owner`	The requesting agent is not owner of the specified model.
Failure Reasons	`not-exists`	The UPA does not maintain the specified user model.

8.3.3.3. register

Supported by	fipa-ups	
Description	The requesting agent wants to register as a consumer of the specified user model using either a `request` CA or a `cfp` CA. In both cases, it suggests privacy conditions. In case of a request, the UPA can only agree or refuse, while in case of a call for proposals, the UPA can make a counter-proposal with a different privacy condition.	

Content	`fipa-ups-um-description`
	
	`fipa-ups-access-condition`	

FIPA Protocol	`fipa-request`
	
	`fipa-iterated-contract-net`	

Example	`(cfp :sender user-agent`
	`:receiver UPA`
	`:content (action UPA`
	`(register`
	`(:user-id joe`
	`:um-type vCARD)`
	`(:access-permission fipa-ups-readwrite`
	`:accessor-description`
	`(:df-description`
	`(:ownership fipa.org)))))`
	`:language SL`
	`:protocol fipa-iterated-contract-net`
	`:reply-with ume-req3`
	`)`	

Refuse Reasons	`sender-unacceptable`	The UPA does not accept the requesting agent as consumer of the specified user model, possibly due to the suggested privacy conditions.
Failure Reasons	`not-exists`	The UPA does not maintain the specified user model.

8.3.3.4. set-access-control

Supported by	fipa-ups	
Description	Modifies privacy regulations concerning a given user model. Only the owner of a user model is allowed to request that action.	

Content	`fipa-ups-um-description`
	
	`fipa-ups-access-condition`	

FIPA Protocol	`fipa-request`	

Example	`(request`
	`  :sender user-agent`
	`  :receiver UPA`
	`  :content (action UPA`
	`  (set-privacy-control`
	`  (:user-id joe`
	`  :um-type vCARD)`
	`  (:access-permission fipa-ups-read`
	`  :accessor-description`
	`  (:df-description (:ownership fipa.org)))))`
	`  :language SL`
	`  :protocol fipa-request`
	`  :reply-with ume-req3`
	`  )`	

Refuse Reasons	`not-owner`	The requesting agent does not own the specified user model, hence the UPA does not allow the requesting agent to modify access conditions.
Failure Reasons	`not-exists`	The UPA does not maintain the specified user model.

8.4. fipa-ups-profile Ontology

8.4.1. Overview

The fipa-ups-profile ontology provides two additional actions for write and read access to user models. Both operations require a user model description to identify the user model component to be accessed. The write operation also requires a value to be written into the user model.

Actions of the fipa-ups-profile ontology

Action	**Input arguments**
`write-user-model`	user model description, value
`read-user-model`	user model description

The following table summarizes, how these actions make use of the fipa-ups-um-description parameters (M = mandatory, O = optional):

Action / **Parameter**	`write-user-model`	`read-user-model`
`:user-ID`	M	M
`:um-type`	M	M
`:um-instance`	O	O
`:um-component`	O	O
`:um-language`	O	O
`:um-ontology`	O	O

In the following two subsections, the fipa-ups-profile actions are specified in detail.

8.4.2. write-user-model

Supported by	fipa-ups-profile	
Description	Modifies a given part of the user model according to a given value. If the requesting agent has permission to write the concerned user model (e.g., due to a previous register), the request will succeed. Otherwise, the agent can attempt to obtain a temporary registration by issuing a cfp to the UPA involving a write-user-model action and a temporary access condition.	
Content	fipa-ups-um-description value	
FIPA Protocol	fipa-request fipa-iterated-contract-net	

Example	`(request`
	`  :sender user-agent`
	`  :receiver UPA`
	`  :content (action UPA`
	`  (write-user-model`
	`  (:user-id joe :um-type vCARD`
	`  :um-component fn)`
	`  “JOE”))`
	`  :language SL`
	`  :protocol fipa-request`
	`  :reply-with ume-req3`
	`  )`	

Refuse Reasons	not-acceptable	The requesting agent is not allowed to write the specified user model part.
Failure Reasons	not-exists	The UPA does not maintain the specified user model.

8.4.3. read-user-model

Supported by	fipa-ups-profile	
Description	Reads a given part of the user model. If the requesting agent has permission to read the concerned user model (e.g., due to a previous registration), the request will succeed. Otherwise, the agent can attempt to obtain a temporary registration by issuing a cfp to the UPA involving a read-user-model action and a temporary access condition. In both cases, the UPA reacts to a successful read request by returning the requested value in a final inform CA.	
Content	fipa-ups-um-description	
FIPA Protocol	fipa-request fipa-iterated-contract-net	

Example	(request
	  :sender user-agent
	  :receiver UPA
	  :content (action UPA
	  (read-user-model
	  (:user-id joe :um-type vCARD
	  :um-component fn)))
	  :language SL
	  :protocol fipa-request
	  :reply-with ume-req3
	  )
	
	(agree
	  :sender UPA
	  :receiver user-agent
	  :content (action UPA
	  (read-user-model
	  (:user-id joe :um-type vCARD
	  :um-component fn :um-language XML)))
	  :language SL
	  :protocol fipa-request
	  :in-reply-to ume-req3
	  )
	
	(inform
	  :sender UPA
	  :receiver user-agent
	  :content “<fn> Joe </fn>”)
	  :language XML
	  :protocol fipa-request
	  :in-reply-to ume-req3
	  )	

Refuse Reasons	not-acceptable	The requesting agent is not allowed to read the specified user model part.
Failure Reasons	not-exists	The UPA does not maintain the specified user model.

8.5. fipa-ups-learning Ontology

8.5.1. Background

Learning is a fundamental attribute of agenthood (Franklin and Graesser, 1996). Even in popular conception, the capacity of a machine to learn is seen as the distinction of a "true" agent from mere software programs. This popular conception is also rapidly becoming a commercial requirement; more and more agent-based applications, from network management to electronic commerce, are including learning technology, which many vendors are providing. Unfortunately, there is much confusion about the term. In its weakest sense, the term "learning" has been applied to any perceived adaptability such as reordering a list based on frequency of use – or simply, base on most recent use. In the other extreme, learning is perceived as magic that can automatically discover anything by watching everything. In contrast, the fipa-ups-learning ontology of ups is scoped to cover the following:

> Learning requires a stricter technical sense of association, based on a more complete theoretical model from learning theory. This eliminates trivial adaptability or linear accounting. Learning is assumed to include both auto-associative (stimulus-stimulus) and hetero-associative (stimulus-response) components.
>
> In contract to magic, this specification is applicable through an explicit definition of observations and fipa-ups-learning service actions. It does have an observational bias toward unary predicates, discrete events, and some representation of logical associations.
>
> There is no singular representational bias. Various techniques such as case-based, neural network, decision tree, and other well-known methods can be used.
>
> This is an agent-based learning service, which raises special requirements not generally met by all learning technologies, generally speaking. For instance, agent-based learning is assumed to be incremental and non-parametric. More particularly, this agent-based learning is directed toward user modeling as the basis for these special requirements.

The user model is fundamentally individual. While several agent-based learning technologies use clustering techniques for "collaborative filtering", the bias of the learning service is toward individual user modeling. Fundamentally, each user should be a cluster of 1 for truly addressing individual differences in taste. There are many application areas for which stereotyped assignment to a group, reliance on the Law of Large Numbers, and tweaking of an appropriate parameters (such as the right number of clusters) to get good segmentation is either inappropriate or impossible.

Collaboration of learned user models is based on this fundamental individualism. The approach is toward sharing and comparing individual models (subject to privacy), with heavy use of multi-agent services and the multi-agent paradigm.

User modeling will provide the prototypical use cases of this service, but this modeling can be conceived as more that preference profiling. Such uses are mentioned in the FIPA97 applications such as for travel, scheduling, and audio-visual preferences, but in other business domains such as manufacturing, financial advising, and clinical decision support, this service is better described as task profiling and includes added requirements such as for sequence learning. Most generally, this service is applicable to the area now called "knowledge management" within industries and variously known as "federated learning" or "collaborative reasoning" within the context of multi-agent systems (Weiss, 1997). Again, collaboration across organizational knowledge is possible, but from the foundation of open, distributed, and individual learning agents.

8.5.2. Formal Overview

8.5.2.1. Observations

The elements of an observation include a state and a label. State is a vector of attribute-values containing the context, objects, results, and/or arguments in observing some label. The label is the name of the discrete event, action, or categorization of the observation.

For example, a product can be defined as a set of features, which would constitute the vector of attribute-values. Given this product representation, an event such as "buy" can be observed by the agent and associated with the features. As another example, a workflow activity can also be defined as a vector of attribute-values. Given such an activity description and the observation of a user's delegation or routing of activities to other users, the user's names can be used as the labels. The agent would observe both the activity and routing to learn their associated contingencies.

8.5.2.2. Associations

The service supports the representation of two types of associations, based on Learning Theory terminology.:

Association between states and labels -- This is typically called hetero-associative or stimulus-response learning. Given a state, the service can predict a label based on past observations.

Association between attribute-values within states -- This is typically called auto-associative or stimulus-stimulus learning. Given a partial state, the service can pattern match to complete the other parts of the state based on past observations.

This distinction is also known within the field of Decision Analysis and Decision Making, particularly in commerce:

Choice tasks – Given some context and set of alternatives, a user is asked to select or choose one (or more) from the set. For instance, among all the offerings in a wine shop, a customer must choose one or more for the needs of a dinner. This is quite literally, picking a "label", given the circumstances.

Matching task – Given a set of attribute-values, match the appropriate value of some other attribute. For instance, the wine shop customer may judge the price of the label too high or too low, given the attributes and experience with other similar wines.

See almost any text in the Psychology of Learning or machine for further descriptions of learning as association. See Hauser & Warnerfelt (1990) for these decision making distinctions in Marketing Science.

8.5.2.3. Representational Bias

This learning service has no implementational bias. Its interface might be provided by case-based reasoning, decision-trees, neural network, or other techniques. On the other hand, it does assume a slight representational bias within the space and distance metrics. As an example of a typical space, see Kanerva's Sparse Distributed Memory for a description of Boolean Space. The service interface is not necessarily limited to $\{0,1\}^n$, a vector of n Boolean attributes, but this space represents the bias toward observations as vectors and distance as a measure of prediction and pattern matching.

A new observation has a distance to a set of past observations. This distance represents the degree to which the new observation belongs or is a probable member of the past observation set. The new observation might be a past observation; it might even be a prototypical past observation if its location in space is central to many/most of the past observations. The new observation may be relatively close or antithetical to past observations, again as a matter of distance in the space.

This degree of similarity or belonging is called "relevance". As a separate more statistical measure, the service also provides a measure of "competence". This latter measure is an estimation of the statistical power and significance of a set of observations. However, this FIPA98 specification does not demand any particular space, metric, or statistical test (aside from being apparently non-parametric).

This learning service is an inferential service, in the sense that it is not a database service. It adopts the computational learning theory of probably approximately correct (PAC) in its internal embedding of observations into its storage. It might perform like a database service under certain conditions, but this is not guaranteed. For instance, a small case-base can provide perfect recall of the nearest match, but larger loads force more heuristic

approximations.

8.5.2.4. Qualities of Service

Performance tracking is an important dimension, which a service might provide for its clients. However, performance can be objectively measured by the client itself; therefore, this learning service does not specify performance in order to remain minimal.

The client is responsible for use and interpretation of the service measures of relevance and competence. Hybridization with a rule-based system would allow for goal-orientation, and sensitivity to application issues. For instance, only the application client can know and control responses to the various contexts – from simple menu ordering to missile launching.

8.5.3. Levels of Autonomy

Fundamentally, FIPA holds that every agent is owned by some user and is therefore responsible for that agent. The severity of this relationship between the agent and user is because of uncertainties in the learning agents. This learning service is very closely aligned with human-agent interaction. The dialog specification will typically be used along with the learning service. This joint use will be mentioned below within several interaction protocols.

Level of Autonomy is a critical agent-user issue. The user is fundamentally responsible but the user must trust and control the agent. For instance, the user may not give the agent any autonomy, in which case a learning agent might simply predict/suggest actions for the user to take. But to the degree that the agent is confident in a prediction, which surpasses some threshold level of trust, the agent can be allowed to take automatic action. Otherwise, it makes suggestions to the user. As the agent gets smarter and smarter, the user can gradually see its performance and allow it more autonomy. Other techniques such as the user telling the agent to delay automation for sometime – while notifying and allowing the user to alter the action – should also be considered.

8.5.4. Grammar of fipa-ups-learning

```
Action              =   memorize
                    |   forget
                    |   choose
                    |   scope
                    |   match
                    |   get-relevance
                    |   get-competence
                    |   get-sensitivity
                    |   get-association
                    |   consult

state               =   "(state" AttributeValue* ")"

AttributeValue      =   "(" SLPredicateSymbol  SLConstant* ")" | "(" SLPredicateSymbol
                    NumericalRange ")"

NumericalRange      =   "<" NumericalConstant
                    |   ">" NumericalConstant

label               =   Word

label-filter        =   label+

competence-filter   =   NumericalConstant "%"
                    |   null

get-relevance       =   " (relevance" NumericalConstant
                    |   NumericalConstant "%" ")"

get-competence      =   " (competence" NumericalConstant "%)"

get-sensitivity     =   " (sensitivity"  "(" AttributeValue NumericalConstant  ")" *
                    ")"

get-association     =   " (association” StringLiteral+ ")"
```

```
consult             =   " (opinion" ((user-id | "anonymous") |   label*)* ") "

normalize-option    =   "as-count"
                    |   "as-cercent"”
                    |   null

sort-option         =   "by-polarity"
                    |   "by-magnitude"
                    |   null

match-option        =   "nearest"
                    |   "prototype"
                    |   null

consult-option      =   "experts"
                    |   "like-me"
                    |   null

number-requested    =   NumericalConstant
                    |   null

user-id-filter      =   user-ID+
```

The grammar for user-ID, um-component, um-type are all attributes of the fipa-ups-um-description defined in the fipa-ups- section of Human Interaction. SLPredicateSymbol, SLConstant, StringLiteral, NumericalConstant, AttributeValue, and Word are further defined in FIPA97, Agent Communication Language.

8.5.5. Actions of fipa-ups-learning

The user model of the fipa-ups-learning ontology must support the following actions:

Action	**Write**	**Read**	**Measure**	**Explain**	**Collaborate**
memorize	X				
forget	X				
choose		X			
scope		X			
match		X			
get-relevance			X		
get-competence			X		
get-sensitivity				X	
get-association				X	
consult					X

Table of fipa-ups-learning Actions

To summarize this list, actions of the fipa-ups-learning consist of the following five action groups as shown. These actions are grouped and labeled only for this presentation; they have no formal significance beyond the actions.

Write – Write and erase are nominally recast as *memorize* and *forget*. Memorize associates the elements of an observation. Forget disassociates the elements. Associate and disassociate are also synonymous terms as used in Michalski's general theory of learning.

Read – In contrast to reading a simple fact as in read-user-model, "reading" from a learned user model can be more complex, and allowing various types of request. For instance, the reading actions are of two types: *choose*, which is hetero-associative, and *match*, which is auto-associative. Within the scope of an um-component, choose finds the strongest association to a label, given a state. Given a partial state and label, match finds the strongest associations to other parts of the state (based on past observation). These actions are identical to the terms used in economic decision theory; these are the two fundamental types of decision-making tasks. For instance, a consumer chooses one product from many or matches price to the intrinsic attributes of a product. Scope is included as a more sophisticated form of choose, as will be described.

Measure – For reading deeper into the representation, fipa-ups-learning provides actions for reading the degree of association. Relevance is used to read a metric of closeness or membership between a state and all states written to a particular label. Competence is used to read the ability or confidence of the fipa-ups-learning service to answer a query.

Explain – Sensitivity and association provide deeper query of the learned knowledge based, which can be used, for explanation and discovery. Sensitivity provides a linear representation of indicators and contra-indicators within a state (although a non-linear analysis is assumed). Association provides representation of the deep associations within the knowledge base.

Collaborate – The consult action is useful when the individual model is insufficient from either a normative or subjective perspective. For instance, when the competencies of the individual and individual user model are not strong enough to make a choice, other opinions can be requested. Two forms of consultation are specified, expert and like-me. In situations where there is a best or right answer, consultation should be based on experts' opinions. When there is not a right answer (when choice depends on individual preference), consultation should be based on the opinions of similar others.

All of these actions are formally described below.

8.5.5.1. memorize

Supported by	fipa-ups-learning
Description	Store an observation in memory, by associating some state or conditions to a particular label. This label might be a category assignment or a consequent action. For instance, when a user files a form into a folder, the attributes of the form can be associated to the folder name. Or if a user assigns a workflow activity to some resource/participant, the activity can be associated to the resource name. The observation of such actions under such conditions is stored. The state is represented as an attribute/value vector.

Content	`fipa-ups-um-description`
	
	`label`
	
	`state`

FIPA Protocols	`fipa-ups-recall-selection`
	
	`fipa-ups-proposal-selection`
	
	`fipa-ups-feedback`
	
	`fipa-ups-sequence`

Example	`(request`
	`  :sender user-agent@iiop://…`
	`  :receiver purchase-agent@iiop://…`
	`  :content`
	`  (action learning-agent`
	`  (memorize`
	`  (:user-id joe-buyer`
	`  :um-type P3P`
	`  :um-component vendors.purchase.feedback)`
	`  VendorABC`
	`  (State`
	`  (parttype A2345)`
	`  (quality 100)`
	`  …`
	`  (QoS wrongPartType)`
	`  )`
	`  )`
	`)`
	`  :language SL10`
	`  :protocol fipa-ups-feedback`
	`  :ontology fipa-ups-learning`
	`)`
	
	This demonstrates a user agent requesting a learning agent to memorize a quality of service issue about a particular order received from a particular vendor (The vendor sent the wrong part type). The state vector of the order (attribute/values) will typically be much richer.
	
	The user is Joe, a buyer who has selected VendorABC. The feedback is memorized at um-component vendors.purchase.feedback.

Refuse Reasons	`Unauthorized`

8.5.5.2. forget

<table>
<tr><td>Supported by</td><td colspan="2">fipa-ups-learning</td></tr>
<tr><td>Description</td><td colspan="2">Erase the association from memory. Removes a previously associated state at a label.</td></tr>
<tr><td>Content</td><td colspan="2"><code>fipa-ups-um-description</code>

<code>label</code>

<code>state</code></td></tr>
<tr><td>FIPA Protocol</td><td colspan="2"></td></tr>
<tr><td>Example</td><td colspan="2"><pre>(request
      :sender user-agent@iiop://…
      :receiver preference-agent@iiop://…
      :content
            (action preference-agent
                    (forget
                      (:user-id mary
                       :um-type P3P
                       :um-compoenent food.vendors.FastFry)
                      TheUsual
                      (state
                        (sandwich DoubleBaconBurger
                         sandwichIngedients <>
                         drinkType Cola
                         drinkSize Large
                         side Salad)
                       )))
      :language SL0
      :ontology fipa-ups-learning
)</pre>This demonstrates the user agent erasing an old case of a fast food order that is being stored as “the usual”. For instance, assume that this order was suggested by preference-agent in a prior call to match (find the typical pattern of order), but the user makes a small change to the order.</td></tr>
<tr><td>Refuse Reasons</td><td colspan="2"><code>Unauthorized</code></td></tr>
<tr><td>Refuse Reasons</td><td><code>Never-memorized</code></td><td>If the given case does not exist, this method has no effect.</td></tr>
</table>

8.5.5.3. choose

Supported by	fipa-ups-learning	
Description	Asks for a choice or list of top choices, given a context state. The request is scoped to the user-ID and um-component of the fipa-ups-um-description. Within this scope, any number of previously observed labels will form the selection set. The service will return choices/suggestions from this selection set. A selection set filter can be specified as label-filter. Otherwise, the selection set is assumed to be all previous labels observed in the um-component. The size of the return list is determined by the number-requested, if specified. Otherwise, the service will return the top choice. All the choices will be returned in rank order of relevance.	

Content	`fipa-ups-um-description`
	
	`number-requested (optional)`
	
	`label-filter (optional)`
	
	`state`	

FIPA Protocol	`fipa-ups-recall`
	
	`fipa-ups-recall-selection`
	
	`fipa-ups-sequence`	

Example	`(query :sender user-agent@iiop://…`
	`:receiver finance-agent@iiop://…`
	`:content (action finance-agent`
	`(choose (:user-id helen-advisor`
	`:um-type P3P`
	`:um-component finance.mutualFunds)`
	`2`
	`(state (age 40)`
	`(risk high)`
	`(goal retirement))))`
	`:language SL0`
	`:reply-with fa-req2`
	`:ontology fipa-ups-learning)`
	
	`(inform :receiver user-agent@iiop://…`
	`:sender finance-agent@iiop://…`
	`:content (label Fidelity Western)`
	`:language SL0`
	`:in-reply-to fa-req2)`
	
	The user-agent asks the financial-agent for two suggested mutual funds, given the user's characteristics and needs (age, risk, goals, etc). The financial-agent is not constrained to a given selection set and therefore selects the top two labels from its experience, most associated with this user's state.	

Refuse Reasons	`Unauthorized`	
Failure Reasons	`Unrecognized-state`	The description of the state is orthogonal to what has so far been memorized. In other words, all measures of similarity are infinite and the service cannot choose.
	`Unrecognized-section`	Um-component cannot be found.

8.5.5.4. scope

Supported by	fipa-ups-learning
Description	This action is similar to choose, except that scope builds a recommendation/consideration set in which three goals are attempted: 1) to present options the agent considers mostly likely to be the best fit, 2) to allow the agent to maximally learn from the selection/rejection process, 3) to allow the user of the agent to explore a broader set of alternatives (as during constructive preferences). In order to achieve these goals the agent should construct a consideration set that is not simply a collection of best items, but that is a set of items that minimizes the redundancy (correlation) between them. By providing a more varied set of items, the agent helps the user learn about the product space and express his/her preferences better during the selection process. In addition, the options rejected by the user in this more varied set help the agent learn much faster about the user's preferences. Otherwise, the content of this action is identical to that of choose. FIPA98 introduces the more advanced quality-of-service provided by this action, but such important directions in learning and personalization are not yet fully explored. For instance, it is expected that injection of novelty, control of the variability, and such could be further specified by the requesting agent. This action is not required by all applications, for which choose may be adequate, but is particular useful for the earlier stages of the user preference learning process and for domains in which the best choice changes over time.

Content	`fipa-ups-um-description`
	
	`number-requested (optional)`
	
	`label-filter (optional)`
	
	`state`

FIPA Protocol	`fipa-ups-recall`
	
	`fipa-ups-recall-selection`

Example	`(query :sender user-agent@iiop://…`
	`:receiver personalprofile-agent@iiop://…`
	`:content (action personalprofile-agent`
	`(scope (:user-id fred`
	`:um-type P3P`
	`:um-component food.wine)`
	`4`
	`(state`
	`(price <20 USD))))`
	`:language SL0`
	`:reply-with wine-req2`
	`:ontology fipa-ups-learning`
	`)`
	
	`(inform`
	`:receiver user-agent@iiop://…`
	`:sender personalprofile-agent@iiop://…`
	`:content`
	`(label "Gallo chardonnay"`
	`"Millstream white zinfandel"`
	`"Kingston State Merlot 1995"`
	`"Ponaine Font-Sane 1995")`
	`:language SL0`
	`:in-reply-to fa-req2`
	`)`
	
	The user agent asks Fred's personal profile agent for a selection of 4 wines, given the

	constraint of price. The agent scoped one inexpensively-priced white, one moderately-priced white, one inexpensively-priced red, and one moderately-priced red. By observing the selection of the user, the agent can learn about the user's preferences for wine type (red or white) and price range (inexpensive or moderate).	
Refuse Reasons	`Unauthorized`	
Failure Reasons	`Unrecognized-state`	The description of the state is orthogonal to what has so far been memorized. In other words, all measures of similarity are infinite and the service cannot choose.
	`Unrecognized-um-component`	Um-component cannot be found.

8.5.5.5. match

Supported by	fipa-ups-learning
Description	Reports the best or the nearest state in memory for a label, given a state and particular label. The given state may be partial or incorrect in regard to the states in memory already stored at that label. For instance, consumers are often asked to value a product (match a price to a given product's attributes). This pattern completion is assumed to be based on past matches between features and prices. An optional match parameter must be included by the sender to determine whether the search should find the nearest state in memory or the prototype (local optimum) of the given state. The default is Nearest.

Content	`fipa-ups-um-description`
	
	`label`
	
	`match-option (optional)`
	
	`state`

FIPA Protocol	`fipa-ups-recall`
	
	`fipa-ups-recall-selection`
	
	`fipa-ups-proposal-selection`

Example	`(query :sender schedule-agent@iiop://…`
	`:receiver preference-agent@iiop://…`
	`:content (action preference-agent`
	`(match (:user-id fred`
	`:um-type P3P`
	`:um-component references.meetings)`
	`Internal`
	`Best`
	`(state (size 10)`
	`(attendees Joe Mary)`
	`(subject ProjectX)`
	`(day Wednesday)`
	`?time`
	`?room`
	`?a/v )))`
	`:language SL0`
	`:reply-with meet-req2`
	`:ontology fipa-ups-learning`
	`)`
	
	`(inform`
	`:receiver schedule-agent@iiop://…`
	`:sender preference-agent@iiop://…`
	`:content (state`
	`(time 14:00) (room R214) (a/v NULL))`
	`:language SL0`
	`:in-reply-to meet-req2)`
	
	Assume that a schedule-agent has received a request to schedule a meeting for Fred, but that the request was not fully specified. It asks the preference-agent to complete the pattern based on what was requested. The preference-agent returns intelligent defaults. The schedule-agent is responsible for scheduling or checking with Fred depending on its policy (and the confidence of these defaults if requested).

Refuse Reasons	`Unauthorized`	
Failure Reasons	`Unrecognized-state`	The description of the state is orthogonal to what has so far been memorized.
	`Unrecognized-section`	Um-component cannot be found.

8.5.5.6. get-relevance

Supported by	fipa-ups-learning
Description	Membership or distance of the state to other states memorized at the label. A large positive value indicates that the state is prototypical. A large negative value indicates that the state is antithetical. A weak positive value indicates that the state is probably known to be associated with the label (but not strongly), while a weak negative value indicates distance from possible membership, to the degree of negativity. For instance, a robin is a prototypical bird. An elephant is certainly not. An ostrich is a bird, but is a borderline case. A platypus is not a bird, but has some attributes making it similar. Normalize-option can be used to request relevance as a raw number of Count, a measure of how many memorized states are relevant. Relevance can also be requested as Normalize, between 100+ for the prototype and –100 for its antithesis. Normalize is the default.

Content	`fipa-ups-um-description`
	
	`label`
	
	`normalize-option (optional)`
	
	`state`

Example	`(query :sender user-agent`
	`:receiver learning-agent`
	`:content (action learning-agent`
	`(get-relevance`
	`(:user-id john`
	`:um-type P3P`
	`:um-component email.folders)`
	`trash`
	`(state <AttributeValues>)))`
	`:language SL0`
	`:reply-with folder-req1`
	`:ontology fipa-ups-learning`
	`)`
	
	`(inform :sender learning-agent`
	`:receiver user-agent`
	`:content (relevance +90)`
	`:in-reply-to folder-req1`
	`)`
	
	This demonstrates a user agent asking whether a set of keywords in an email item is probably like those labeled “trash” (assuming prior items have been memorized as trash). The learning agent responds with a strong positive indication.

8.5.5.7. get-competence

Supported by	fipa-ups-learning
Description	The statistical significance of the fipa-ups-learning service in answering a relevance query. This measure is sensitive to the service's maturity (number of observations) and clarity of patterns across observations (significance). The statistical significance of the entire association at the label is returned. This represents the competence to answer questions about the label in general. However, if a state vector is optionally provided, the competence level is scope to those associations only within the state.

Content	`fipa-ups-um-description`
	
	`label`
	
	`state (optional)`

Example	`(query :sender user-agent`
	`:receiver learning-agent`
	`:content (action learning-agent`
	`(get-competence`
	`(:user-id john`
	`:um-type P3P`
	`:um-commponent email.folders)`
	`trash))`
	`:language SL0`
	`:reply-with folder-req2`
	`:ontology fipa-ups-learning`
	`)`
	
	`(inform :sender learning-agent`
	`:receiver user-agent`
	`:content (competence +10%)`
	`:in-reply-to folder-req2`
	`)`
	
	This demonstrates a user agent asking about the learning agent's competence to answer queries about "trash". The learning agent responds with weak competence, such as if it had little experience or clear associations.

8.5.5.8. get-sensitivity

Supported by	fipa-ups-learning
Description	The change in output relevance caused by changing the input state of the sensor. If the sensitivity is positive, then the reverse of the current sensor signal would result in increasing relevance. If the sensitivity is negative, then the reverse of the current sensor signal would result in decreasing relevance. The sensitivity value should range from 100 (reverse the signal) to -100 (keep the signal). The fipa-ups-learning assumes that the representation is non-linear; therefore, the calling agent should beware of possible nonlinear effects. In other words, the sensitivity of one sensor is within the context of all other sensor states. Changing one sensor state can effect the sensitivity of other sensors in unexpected ways. This nonlinearity can be further explored by query of *association.* However, *sensitivity* provides a simpler, linear presentation of causality – even if the deeper knowledge is more complex. A query of multiple sensors' sensitivity can be returned in different sort-options as requested. The sensitivities can be sorted by Polarity (most positive to most negative) or by Absolute magnitude (most powerful to least powerful). Polarity is the default.

Content	`fipa-ups-um-description`
	
	`label`
	
	`sort-option` (optional)
	
	`state`

FIPA Protocol	

Example	`(query :sender george-agent@iiop://…`
	`:receiver susan-agent@iiop://…`
	`:content (action susan-agent`
	`(getSensitivity`
	`(:user-id susan-pediatrician`
	`:um-type P3P`
	`:um-component practice.prescription)`
	`Septa`
	`Absolute`
	`(state (age child)`
	`(diagnosis otitis-media)`
	`… )))`
	`:language SL0`
	`:reply-with consult-req2`
	`:ontology fipa-ups-learning)`
	
	`(inform :receiver george-agent@iiop://…`
	`:sender susan-agent@iiop://…`
	`:content (sensitivity`
	`(age child –100%) (diagnosis otitis-media +75%) …)`
	`:language SL0`
	`:in-reply-to consult-req2)`
	
	One physician's agent asks another physician's agent for information about a practice pattern. In particular, it gives the state of a patient being presented and asks for the sensitivity of attribute/values, which are indicators and contra-indicators. The agent answers that Septa is contraindicated for children, although generally indicated for the diagnosis.

8.5.5.9. get-association

Supported by	fipa-ups-learning
Description	Learned relationship between attributes/values across all memorized states. No matter what the underlying representation, these relationships must be reported as propositional formulas. While it is possible to extract all associations, the caller will typically ask for associations between specific attributes. This is accomplished by use of state; this service will provide the associations between the given attribute-values. If state is not specified, then all associations will be returned. If only one attribute-value is specified, then all associations to other attribute-values will be returned. Otherwise, only those attribute-values within the state will be returned – as a clique of associations. The caller can also filter based on a level of competence. Only those associations that are stronger than this filtering will be returned.

Content	`fipa-ups-um-description`
	
	`label`
	
	`competence-filter` (optional)
	
	`state` (optional)

Example	`(query :sender seller-agent@iiop://…`
	`receiver user-agent@iiop://…`
	`:content (action user-agent`
	`(get-association`
	`(:user-id user-agent`
	`:um-type P3P`
	`:um-component preference.books)`
	`purchases))`
	`:language SL0`
	`:reply-with preference-req`
	`:ontology fipa-ups-learning)`
	
	`(inform :receiver seller-agent@iiop://…`
	`:sender user-agent@iiop://…`
	`:content`
	`(association`
	`“((subject astomomy) and (40 USD < price < 70 USD))“`
	`“((subject scifi) and (author A) and (type soft-cover))“`
	`“((subject scifi) and (author B) and (type hard-cover))“)`
	`:language SL0`
	`:in-reply-to preference-req`
	`)`
	
	A seller agent can ask a user directly for its associative rules for book preferences. The user-agent responds with three conjunctive rules, which can be used as a database query of the seller’s catalog. The rules that are returned are treated as terms.

8.5.5.10. consult

Supported by	fipa-ups-learning
Description	Searches for the appropriate recommendations based on knowledge gathered from others. The service returns recommendations (opinions). Opinions include the source of the recommendation and the label (action, choice, predicted event, etc). The number of opinions requested can by specified by the user-id-filter. The particular users to search for can also be specified by user-id-filter. Each service implementation can combine these options as it sees best for its quality of service. The consultation can be one of two types. First, experts can be requested by use of the consult-option, expert. The service will search for models that are most competent to answer the query. Otherwise, the consult-option, like-me, can be used to search for models based on similarity to the given user-id. Finally, consultation can be relatively unspecified by a particular label, in which case, the service will provide the recommendations of others. One the other hand, if the label is specified, the service will provide evaluations of the label. In this latter case, the search is more discriminating, based on other's competence with the particular label.

Content	`fipa-ups-um-description`
	
	`user-id-filter (optional)`
	
	`number-requested`
	
	`consult-option (optional)`
	
	`label (optional)`
	
	`state`

FIPA Protocol	

Example	`(query :sender mary-agent@iiop://…`
	`:receiver group-agent@iiop://…`
	`:content (action group-agent`
	`(consult (:user-id`
	`:um-type P3P`
	`:um-component diagnosis.cancer.breast)`
	`5`
	`(state <>)))`
	`:language SL0`
	`:reply-with diagnosis-req12`
	`:ontology fipa-ups-learning)`
	
	`(inform :receiver mary-agent@iiop://…`
	`:sender group-agent@iiop://…`
	`:content (opinion (jim-radiologist negative-indication)`
	`(sara-oncologist negative-indication))`
	`:language SL0`
	`:in-reply-to diagnosis-req12)`

Failure Reasons	`No-consultants`
	
	`No-sharing`

As with the FIPA ACL, this list of actions is intended to be minimal set. For instance, along with relevance and competence, performance of the service might also be important. However, performance – the degree to which the agent recommendations agree with actual user behavior – can be measured by the client agent as an observer. Such tracking would be a good convenience provided by the service itself, but the action list presented here is limited to those computations that can be performed by and only by fipa-ups-learning ontology.

8.5.6. Interaction protocols of fipa-ups-learning

Much like the FIPA ACL communicative acts are only speech primitives and need to be structured into protocols to show application value, these actions of the fipa-ups-learning service also are only the primitives and need typical patterns of usage together to be defined. Normative protocols will not be included in FIPA98, but as FIPA moves toward protocol and communication control, a better formal foundation will be established to define such protocols for learning. For instance, the specification of protocol nesting will probably be required, such as when fipa-ups-learning is used within the context of ContractNet bid selection.

The use of fipa-ups-learning will be obvious enough to some readers, but this specification also has an onus to describe the application use of such actions. The protocols should raise the semantics of fipa-ups-learning to the level of application uses. Future protocols should provide the details, but the following descriptions are indicative:

Semiotic sequence – Observations are chained so that the service represents a learned state-transition predictor. Such sequence learning is semiotic as a pattern-matching method, which can generate novel but reasonable sequences to observed states never seen before.

Proposal selection – Within the context of Contract Net or FIPA's auctioning protocols, fipa-ups-learning protocols can describe how to select a best proposal (choose) or valuate/evaluate some element of it (match) given all the other elements. The collection and learning of feeback data at the conclusion of ContractNet is also possible. This solution is applicable to vendor selection, process routing, and a number of other similar activity-resource matchings.

Facilitated consultation – While the fipa-ups-learning action, consult, allows for search of other opinions within the context on one service, a federate system of fipa-ups-learning services is also possible. For instance, user-agents or learning services can register competence levels as a basis for search. One agent queries for competent agents through such directory facilitation. As it itself receives advice and takes its own actions (and/or gets feedback), it additionally registers or updates its competence level for other agents to locate.

Need elicitation – Electronic commerce scenarios in which the user needs to dialog with a seller agent will need both user dialog management and learning. For instance, while the goal is to scope and choose one or more items for purchase, questions and answers about the product space and the user's needs can be captured in the state and explored with match and get-sensitivity, such as looking for the unknown but strongest indicators and contra-indicators.

Much more generally, user dialog management can and should be used together with fipa-ups-learning for a variety of situations. Many issues of user controllability, trust, and mixed-initiative are most critical when learning agents are considered. In this critical sense, protocols will be needed to capture the best methods of user interaction with an agent also using such a learning service.

8.5.7. References

Franklin and Graesser, 1996. Is it an agent or just a program?
http://www.msci.memphis.edu/~franklin/AgentProg.html.

Distributed Learning Webliography. http://dis.cs.umass.edu/research/agents-learn.html

Weiss, G., 1997. Distributed Artificial Intelligence Meets Machine Learning: Learning in Multi-Agent Environments. Springer

Slovic, Paul. The Construction of Preferences. American Psychologist. May 1995, p. 364-371.

Annex A
(informative)

Examples

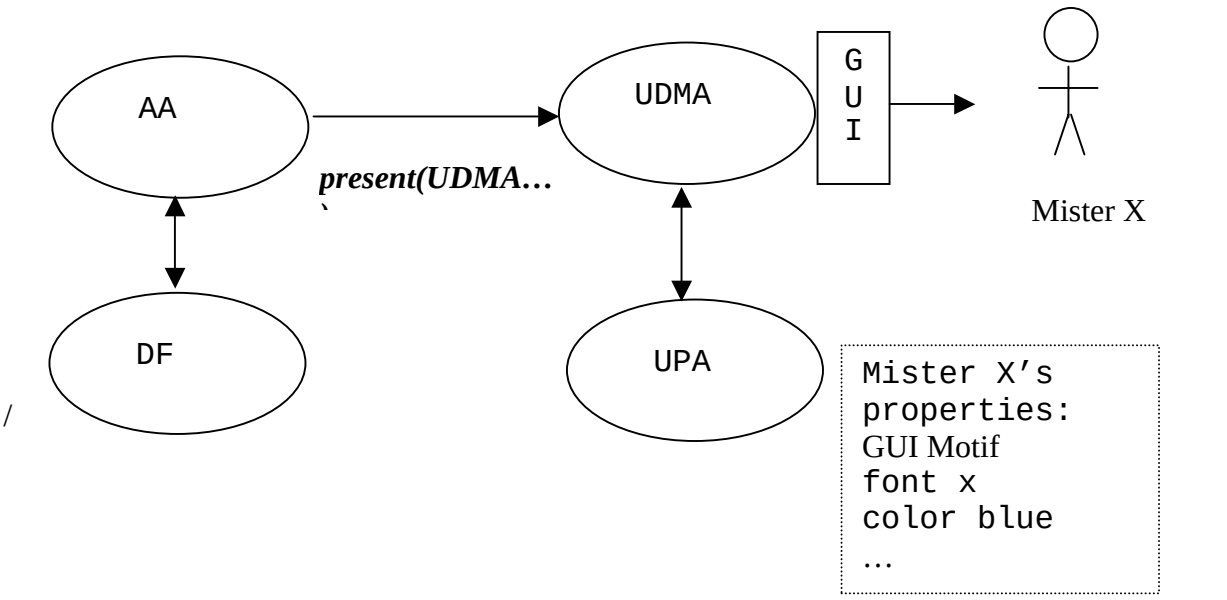
The combination of the UDMA and the UPA can support a number of the functions needed by user-agents and application agents when communicating with the human user. In many previous multi-agent systems the roles of the user-agent, UDMA and the UPA have been supported by one agent usually referred to as the user-agent. Sometimes separation between the GUI and the user-agent is modelled by a user-agent and user-interface agent. We draw upon a number of multi-agent applications to illustrate where the operation of the UDMA is located within an architecture.

A.1 Example 1

A.1.1 From Agent world to Human

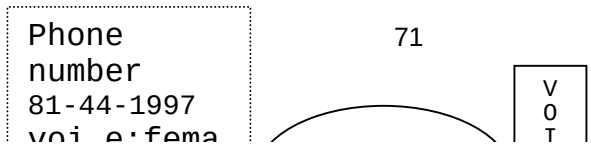
Case 1 UDMA interact with UPA

present(UDMA font x Color blue…)



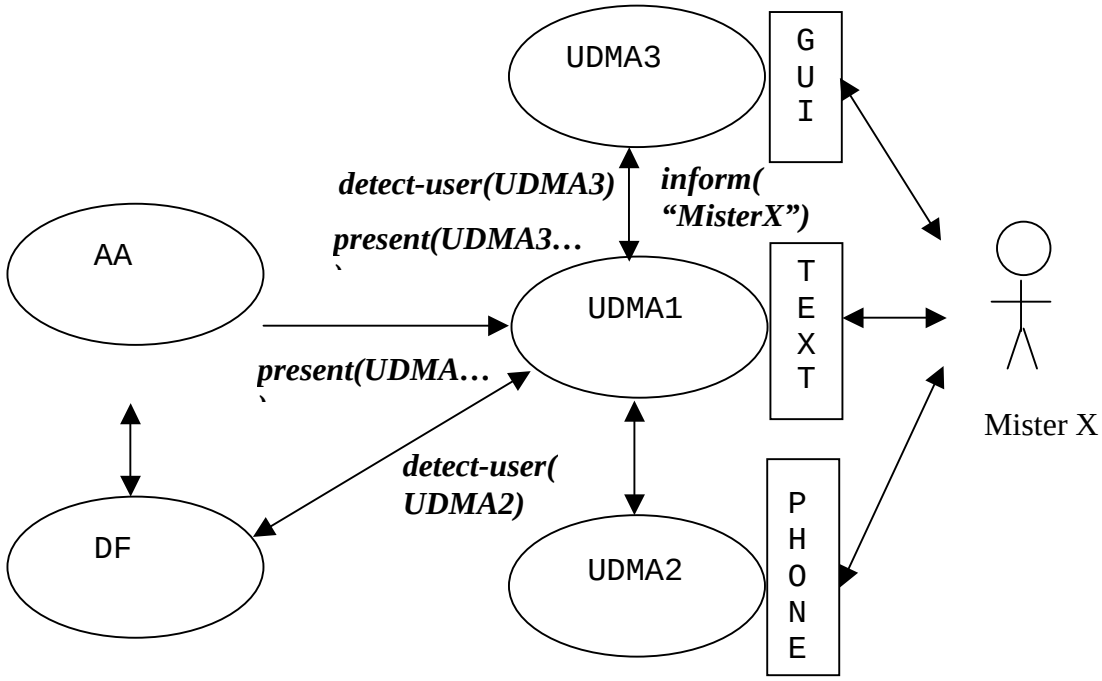
1) AA ask DF for UDMA (Mister X)
2) DF return UDMA (name, address…)
3) AA send to UDMA (present UDMA ……)
4) UDMA ask UPA (Mister X's favorite style)
5) UPA return Mister X's properties
6) UDMA contact Mister X with special style.

Case 2 interact with multi UDMA



1) AA ask DF for UDMA (Mister X)
2) DF return UDMA (name, address…)
3) AA send to UDMA (present UDMA ……)
4) UDMA ask UPA (Where is Mister X)
5) UPA return location of Mister X (Mister X is not login)
6) UDMA ask UPA (How to contact Mister X)
7) UPA return UDMA (Voice or Phone is available)
8) UDMA ask DF (UDMA with Voice, or Phone)
9) DF return UDMA(Voice: name, address, Phone:name address)
10) UDMA contact to New UDMA (Voice or Phone)
11) New UDMA(2 or 3) contact Mister X

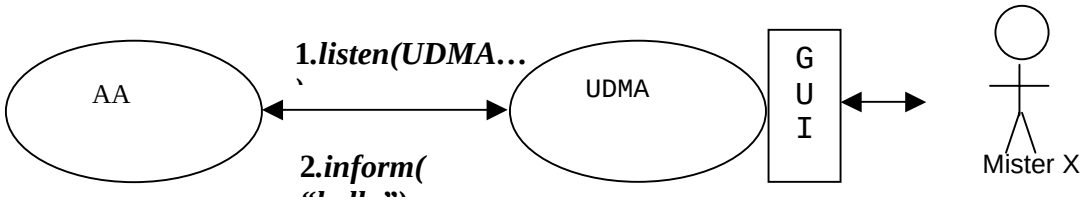
Case 3 UDMA try to detect user



1) AA ask DF for UDMA (Mister X)
2) DF return UDMA1 (name, address…)
3) AA send to UDMA 1(present UDMA1 ……)
4) UDMA1 ask DF (which UDMA can detect Mister X?)
5) DF return UDMA2,3 (name, address, ….)
6) UDMA1 send UDMA2,3 (detect-user (Mister X))
7) UDMA3 return message(inform Mister X is here)
8) UDMA1 send UDMA3 (present UDMA3 ….)

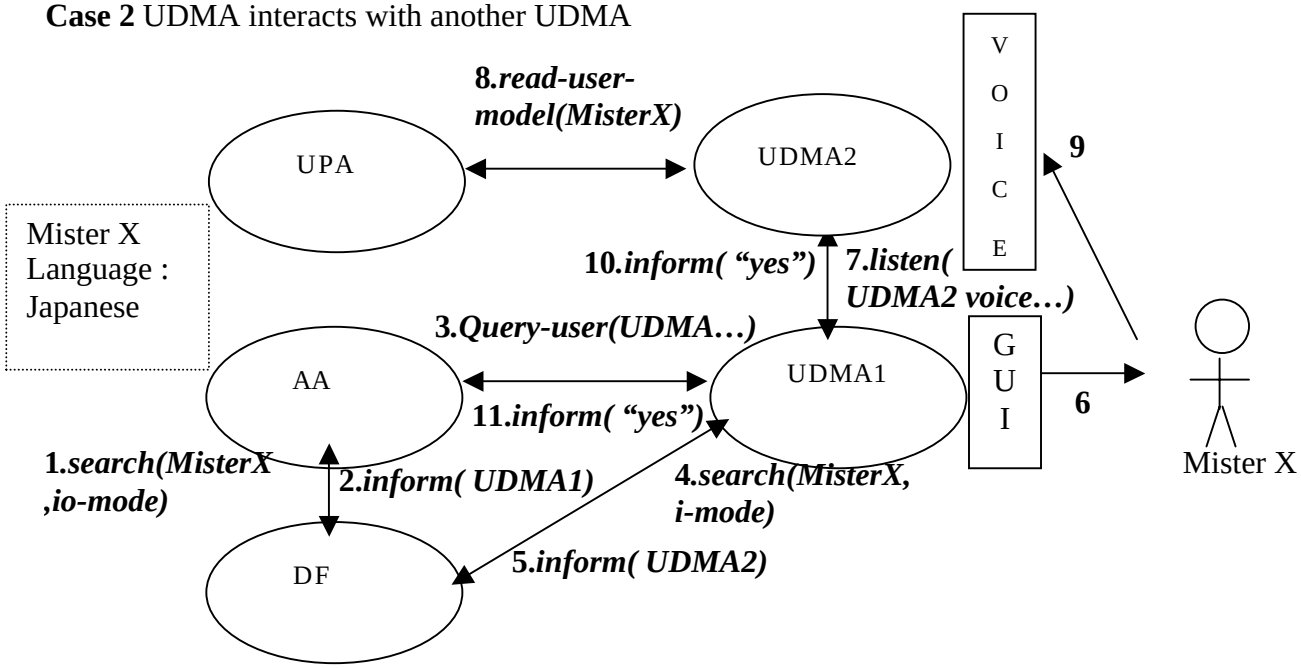
A.1.2 From Human to Agent

Case 1 UDMA listens to user (Simplest)



1) AA request to UDMA (listen UDMA ……)
2) UDMA listen Mister X's action.

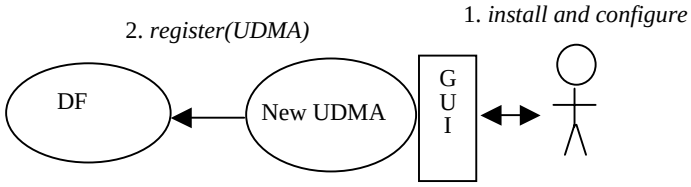
Case 2 UDMA interacts with another UDMA



1) AA ask DF(Mister X, io-mode)
2) DF return UDMA1 (name, address…)
3) AA send to UDMA1 (query-user UDMA ……)
4) UDMA1 ask DF(Mister X, i-mode) to search more efficient input agents.
5) DF return UDMA2
6) UDMA1 present Mister X by Graphical message(Output only).
7) UDMA1 requests UDMA 2(listen UDMA2)
8) UDMA2 asks UPA, which is found through DF, (read-user-model) to get his language for its voice recognition engine.
9) UDMA2 listen Mister X
10) UDMA2 return message to UDMA1
11) UDMA1 return message to AA

A.1.3 Install New UDMA

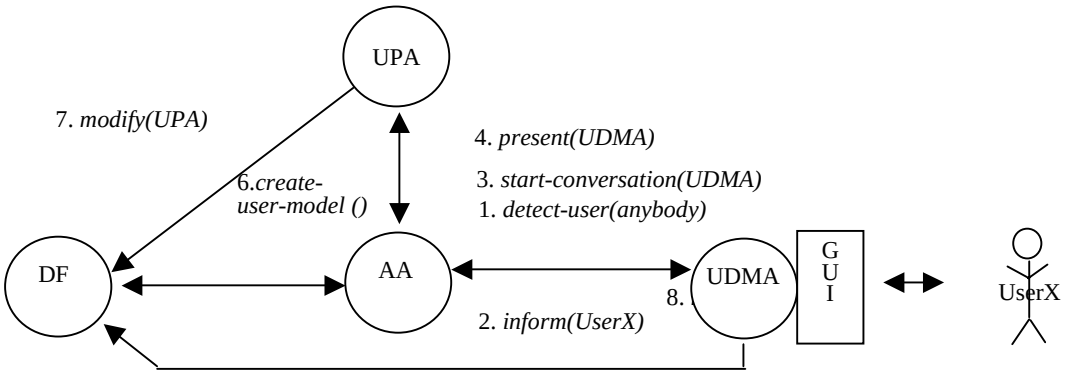
Install new UDMA



1) Install and configure UDMA by User
2) UDMA register to DF (register (UDMA…))

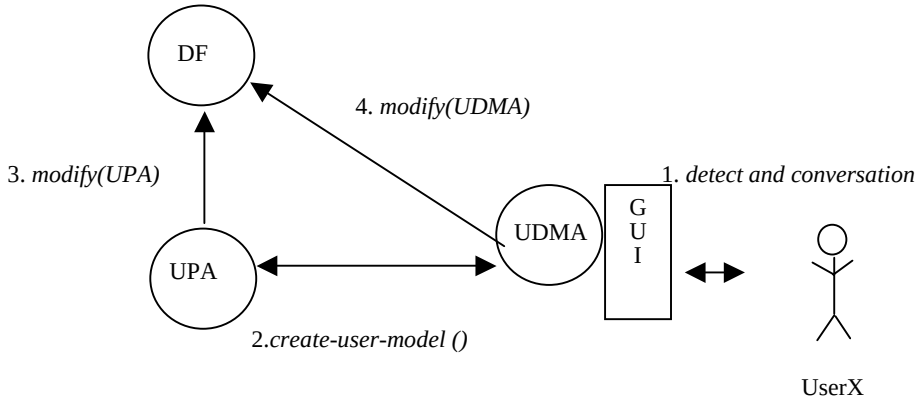
A.1.4 Register New User

Case1 Using Application Agen(AA) to register new User



*AA already knows UDMA and UPA (name,address)
1) AA send to UDMA (detect-user)
2) UDMA return UserX (inform UserX)
3) AA detect the new user (UserX) or AA ask to DF (search UPA(UserX))
AA send to UDMA (start-conversation)
4) AA send to UDMA(query-user(name,phone ,email,etc…)
5) AA send to UDMA (stop-conversation)
6) AA send to UPA (create-user-model(name……))
7) UPA send to DF (modify UPA properties)
8) UDMA send to DF (modify UDMA properties)

Case2 UDMA register new user (UDMA detect new user by itself:)



1) UDMA detect new User and UDMA start conversation to identify user.
2) UDMA send to UPA (create-user-model(name……))
3) UPA send to DF (modify UPA properties)
4) UDMA send to DF (modify UDMA properties)

A.2 Example 2

Many systems have the concept of a *user agent* which acts on behalf of its user, and interacts with the user via a graphical user interface on the user's personal computer. (User agents may inform the user of newly made appointments, of travel plans and reservations, ...). Now suppose the user is not at his/her PC, rather is only reachable by telephone or fax. The user agent can not use its built-in GUI to notify the user of an urgent appointment. Instead of the system designer having to integrate all possible means of interfacing to the user in the user agent, the user agent need only to find an agent able to carry out the required actions. Thus, it might find, through the DF, an agent in the public switching network which can perform speech generation through a telephone as well as another agent which can send faxes. Upon determining which service is most appropriate (according to cost, speed of service, etc.) the agent can select the appropriate UDMA and request it to deliver the notification to the user. (For redundancy purposes, it might even have both UDMAs notify the user!). The UDMA's specified in this document would enable such a scenario described here to be supported.

A.3 Example 3

Part 5 fipa 97

In the Personal Assistant (PA) scenario described in FIPA 97, Part 5, agents act on behalf of the user to schedule appointments. This is all fine if each participant in the meeting has his/her own user agent. However, it may occur that a participant has no personal assistant, rather maintains his/her own calendar. In this case, the PA needs to interact directly with such a participant to schedule the meeting.

One central class of intelligent agents is that of a personal assistant (PA). It 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.

This application has an obvious need for a UDMS in which multiple modalities can be supported. In this way the development of the personal assistant can be concentrated in the area in which it exhibits assistant type functionalities rather than having to understand how to realise a particular content communication with a user. And can benefit from the general services and specification of the UPS in modelling the interests of the user. Similar benefit from such specifications would be encountered by the personal travel assistant.

A.4 Example 4

An effective means of information filtering and retrieval, in particular for digital broadcasting networks, is of great importance because the selection and/or storage of one's favorite choice from plenty of programs on offer can be very impractical. The information should be provided in a customized manner, to better suit the user's personal preferences and the human interaction with the system should be as simple and intuitive as possible. Key functionalities such as profiling, filtering, retrieving, and interfacing can be made more effective and reliable by the use of agent technologies.

Overall, the application provides to the user an intelligent interface with new and improved functionalities for the negotiation, filtering, and retrieval of audio-visual information. This set of functionalities can be achieved by collaboration between a user agent and content/service provider agent. The intelligent interface specification can benefit from the multi UDMA scenario for enabling various communication with the user. In fact many of the requirements defined in this application (UPA, learning requirements etc.) relating to the user interaction with the system have been specified in a general way in the FIPA part 8.

A.5 Example 5

Across the world, numerous service providers emerge that combine service elements from different network providers in order to provide a single service to the end customer. The ultimate goal of all parties involved is to find the best deals available in terms of Quality of Service and cost. Intelligent Agent technology is promising in the sense that it will facilitate automatic negotiation of appropriate deals and configuration of services at different levels.

Part 7 of FIPA 1997 utilizes agent technology to provide dynamic Virtual Private Network (VPN) services where a user wants to set up a multi-media connection with several other users.

The service is delivered to the end customer using co-operating and negotiating specialized agents. Three types of agents are used that represent the interests of the different parties involved:

1. The Personal Communications Agent (PCA) that represents the interests of the human users.

2. The Service Provider Agent (SPA) that represents the interests of the Service Provider.

3. The Network Provider Agent (NPA) that represents the interests of the Network Provider.

The PCA acts on behalf of the user in its dealings with service providers and hence needs to communicate with the human user to inform the user about services etc. The PCA can benefit by exploiting the UMDS and UPS specifications for the human communication requirements and hence focus design and development resources on the negotiation specification with the service providers. The service providers agent will also benefit form these specifications of the UPA services as it will also model the user form a service perspective.

Annex B
(informative)

Translation between End User Forms and ACL Forms

Human agent interaction actively tackles a given situation to form a goal and to set a sequence of actions to reach the goal. This process needs to have the ability to store, retrieve, and manipulate its knowledge. It also should be able to get necessary knowledge and to deliver its processing results upon agent's requests. It entails that human agent interaction should have a method, i.e., a language, for describing and managing those knowledge for internal representation or external communications.

It is desirable to represent knowledge both internal to the agent/human and for external communication by the same, uniform description language. This language must have the following features: a) independence of the types of agents and software, b) ability to represent various information and knowledge, c) to support human/agent and agent/agent communications with no difficulty, and d) to comply with standards.

This language is required to describe and manipulate various objects and knowledge in a uniform fashion and to support the communications between human/agent.

B.1 Purpose

There are various human agent interactions. However, human users can not usually use agent's language directly. We need translation between human and agent/UDMS. We need a uniform way to represent the contents in these processes. There are many situations in human-created contents like natural language, speech utterance, multi-modal action (e.g., pointing device, motion captured by camera, gesture, mouse, etc.), HTML form (or any other SGML-like form), command, and so on.

B.2 Situation of Human/Agent Communication

The followings are exemplary situations in human agent communication:

1. The communication unit between human and agent is called an utterance. The utterance is a sentence, which is made up of a sequence of words. The communication unit can be one of the following: a set of pre-defined messages (primitive), a set of message templates (intermediate), sublanguage (controlled), or full natural language (human-oriented).

2. A process for human agent communication receives an utterance and analyzes (parses) the sentence

3. A process assigns a meaning to the parsed sentence.

4. A process forms a plan to respond to the given utterance, and may co-operate with other human/agent to execute the plan. This co-operation will be represented as a sequence of actions that will be executed by the process. This process requests to execute corresponding actions.

B.3 Representation for Varieties of Contents: Categories of Objects and Knowledge Representation Scheme

Objects that are to be handled by an agent can be physical objects or the knowledge representation of them. Physical objects include text, database entity, drawings or images, audio/visual objects, and objects that have application-specific structures. The knowledge representation can be of four categories of schemes:

1. "Logical representation schemes" use expressions in formal logic to represent a knowledge base. (e.g.. Prolog is an ideal programming language for implementing these schemes.)

2. "Procedural representation schemes" represent knowledge as a set of instructions for solving a problem. (e.g., production systems)

3. "Network representation schemes" capture knowledge as a graph in which the nodes represent objects or

concepts in the problem domain and the arcs represent relations or associations between them. (e.g., semantic networks, conceptual dependencies, conceptual graphs)

4. "Structural representation schemes" extend networks by allowing each node to be a complex data structure consisting of named slots with attached values. (e.g., scripts, frames and objects)

The problems of human agent interaction are mainly related to the problems of searching appropriate objects or knowledge and of applying retrieved knowledge to certain actions. Hence, those objects and knowledge need to be represented in an appropriate structure for the processing in different message levels like primitive, intermediate, controlled and human-oriented. The description language should cope with this issue.

B.4 Conformance to Criteria

This language follows the criteria:

(a) Independence of the types of agents and software

(b) Ability to represent various information and knowledge

(c) To support human agent interaction with no difficulty

(d) To comply with other FIPA standards.

B.5 An Example: A Uniform Description Language for Translation between End User Forms and ACL Forms

We provide a guideline for representing human-agent communication and shared information/knowledge among them, that is based on SGML. We call it here *SKDL* (Structured Knowledge Description Language)

B.5.1 General Structure of SKDL

SKDL is composed of three parts: header (Header), definition part (front) and data part (body). Header has information including ID, name, type, and the definition part prescribes attribute/value pairs that are to be used in the data part. Figure 8 shows a general structure of SKDL.

Figure 9 depicts the detailed structure of the header and the definition part.

```
<fipa-skdl type=service_type>
        <fipa-skdl-Header> [header information] </ ipa-skdl-Header>
        <front> [definition of attribute - value pairs] </front>
        <body> [SKDL elements] </body>
</fipa-skdl>
```

Figure 8 General structure of SKDL

```
<fipa-skdl-Header>
        <fipa-skdl-ID> [ID]        </fipa-skdl-ID>
        <fipa-skdl-Name> [name]        </fipa-skdl-Name>
        <fipa-skdl-Ver> [version number]        </fipa-skdl-Ver>
        <fipa-skdl-Type> [SDKL type]  </fipa-skdl-Type>
        <fipa-skdl-Date> [date and time of creation]  </fipa-skdl-Date>
        <fipa-skdl-Agent> [skdl creation agent]        </fipa-skdl-Agent>
        <fipa-skdl-Note> [explanation]  </fipa-skdl-Note>
</fipa-skdl-Header>
<front>
        <attrdefgroup>
                <attrdef> [definition of an attribute]    </attrdef>
                <attrdef> [definition of an attribute]    </attrdef>
                [repetition of attrdef's]
        </attrdefgroup>
```

```
        <valdefgroup>
            <valdef> [declaration of an attribute value] </valdef>
            <valdef> [declaration of an attribute value] </valdef>
            [repetition of valdef's]
        </valdefgroup>
</front>
<attrdef>
        <attrname> [attribute name] </attrname>
        <attrlist>
            <attr> [real attribute]</attr>
            [repetition of attr's]
        </attrlist>
</attrdef>
<valdef>
        <valname> [value name] </valname>
        <vallist>
            <val> [real value]</val>
            [repetition of val's]
        </vallist>
</valdef>
```

Figure 9 Header and definition part of SKDL

B.5.2 An Example Description of SKDL for Human Agent Communication

Let's assume the following situation to show an example of SKDL description for an application of Personal Travel Agent. A traveler requests the following to the User Interface Agent (UIA) using a telephone call.

> "I must meet M. Kane of Rosebud Company in New York in the afternoon of the 29th of September. Plan me a trip for this appointment. I would like an answer by tomorrow evening, take my standard preferences into account."

The following situations can be described in a uniform way by SKDL.

1. UIA receives the voice of a traveler and passes it to the Speech Recognition software (or agent). Speech Recognition recognizes a natural language sentence and sends it back to the UIA.

```
<fipa-skdl type=NLSen>
        <fipa-skdl-Header> .....
        <body>
            <sen> I must meet M. Kane of Rosebud company in
                  New York in the afternoon of the 29th of
                  September.
            <sen> Plan me a trip for this appointment.
            <sen> I would like an answer by tomorrow evening,
                  take my standard preferences into account.
        </body>
</fipa-skdl>
```

Step1: Natural language sentence as the output of Speech Recognition software (or agent)

2. UIA passes the natural language sentence (result of step 1) to Natural Language (NL) Understanding software (or agent), and then the latter parses the message and returns the meaning (parse tree or concept) to the UIA

```
<fipa-skdl type=PTree>
        <fipa-skdl-Header> .....
        <body>
            <sen-tree>
```

```
                    <agent> I
                    <action>
                        <tense> must
                        <verb> meet
                    <object> M. Kane
                        <of> Rosebud company
                        <in> New York
                    <time> afternoon
                        <of> 29th, September
                </sen-tree>
            .....
            </body>
    </fipa-skdl>
```

Step2: Meaning for the NL sentence as the output of NL Understanding software (or agent)

3. UIA receives the result of step 2 and then passes it to Planning software (or agent). Planning software (or agent) develops a plan for the message and returns it back to UIA.

```
    <fipa-skdl type=plan>
        <fipa-skdl-Header> .....
        <body>
            <plan>
                <action_class> plan_trip
                    <Actor> user
                    <path> PATH
                    <Destination> New York
                <preconditions> ....
                <decomposition> ....
            </plan>
            <plan>
                <action_class> ....
            .....
        </body>
    </fipa-skdl>
```

Step3: Plan for the meaning of the NL sentence as the output of Planning software (or agent)