ESPRIT Project No. 25 338

Work packages D,E,F

Agent Framework

Technical Summary

ID:	Document-Name V. 1.0	Date:	23-Mar-99
Author(s):	SAB	Status:	draft
Reviewer(s):		Distribution:	


FollowMe

Change History

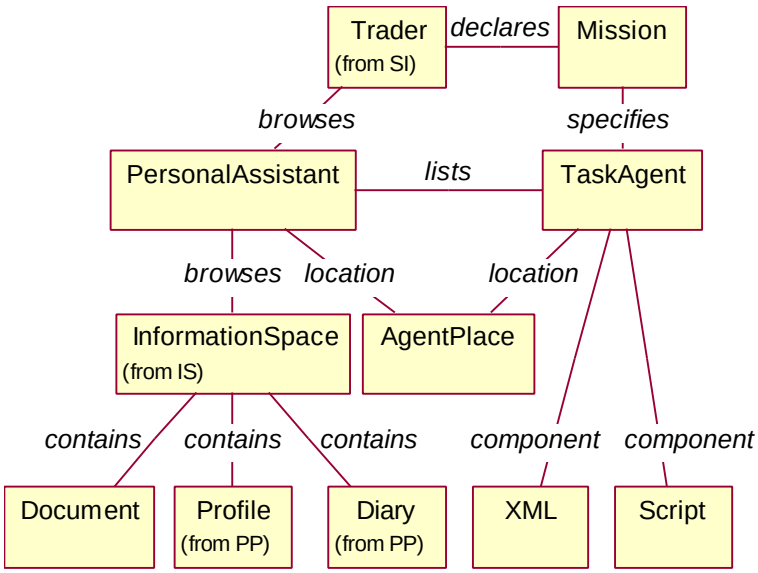
Document Code	Change Description	Author	Date
Technical summary	First version of document.	SAB	23-Mar-99

Agent Framework: Technical Summary

The Agent Framework is composed of three major work-packages. These are the Autonomous Agents work-package which deals with agent functionality, the Personal Profiles work-package for modelling user data, and the Service interaction work package which provides support for agent interaction with services. This technical overview re-iterates the requirements of each work package and covers the principal scenarios actually implemented.

I. Autonomous Agents

The aim of this work-package was to allow users to create and run their own agents, and to manage the use of agent-places, all within the context of the mobile-object work-bench (MOW). A high-level domain model covering the main components of this work is shown below.



Personal Assistant

The PA is the main point of contact for the user, and may be thought of as a kind of personal secretary. The PA is an agent specialised for this purpose alone. The user interface to the PA is separated from the PA functionality itself, enabling alternative interfaces compliant with User Access to be devised. The PA provides access to the user's private information space and to the trader which is used as a mission repository.

Task Agent

The task agent is a general-purpose agent which may be configured by providing it with mission on its creation. The PA maintains a reference to all task agents throughout their lifetimes. TA's may move between agent places as determined by their missions.

Agent Place

An agent place extends the basic idea of place from the MOW. Agent places provide an administrative interface which allows the owner of that place to monitor, and kill, resident agents, and to create Personal Assistants for new users.

Mission

A mission is defined by an XML (eXtensible Markup Language) file and defines the behaviour of a task agent.A mission is composed of a number of textual elements which define the components that make up the agent when it is first created. The remaining domain classes listed below may all be used as mission elements. The mission also defines the mapping rules which associate these mission elements with appropriate Java implementations.

Script

The agent programmer is shielded from the complexities of Java and the MOW by using the JavaScript scripting language (formalised as ECMAScript). To provide support for mobility, the script interpreter was written so that it's entire runtime state could be saved as a serialisable object.

XML

Whereas the aim of the personal-profile work package was to capture generic personal information, there is also the need to represent application specific data within an agent. The extensibility of XML allows new elements to be added to the language as long as they follow a standard markup syntax. The classes that implement this application specific XML present the agent script with a standard programmatic interface to this data (a document object model).

Document

Documents implement the User Access Document interface, and so are the standard components for all communication between the agent and the user. Each document is associated with a set of style rules defined in XSL, which may be applied to the document, rendering it in HTML. Documents support two-way communication using standard HTML form components such as buttons and text inputs. Documents representing different pages may be hyperlinked together to build sophisticated user interfaces.

The main use-cases implemented as part of this work-package are as follows. Additional actors involved are the Users who interact with their Personal Assistant, and the Administrator who looks after an agent place:

Manage agent place

The role of administrator includes starting-up and shutting-down agent places. While an agent place is active, the administrator oversees the agents resident within that place and may opt to kill them if they are acting antisocially.

Create Personal Assistant

Another administrative role is to create Personal Assistants for new users. Each personal assistant is created with its own private information space, where the user may store their personal data and other documents. The PA is registered with the trader so that it may be located by the user later on.

Contact Personal Assistant

Before the user can interact with their agents, they must locate their PA and create an interface to interact with it. This work-package supplies a Java based interface to perform these functions.

Move Personal Assistant

Once contact with the PA is established, the user may move their PA to a different location. This may be required, for example, if the original agent place must be shut-down.

Browse information space

The PA enables the user to search their information space so that they may edit their personal data, or read reports sent back by agents. The viewer presents the information space as a collection of nested folders, and selecting any of the contents will activate the appropriate application.

Launch task agent

Launching a task agent involves browsing the trader for the required mission. Missions may be selected by the user and edited, or launched directly. A new sub-space is created within the PA's own information space which may be used as a work-space by the new task agent. The task agent is also supplied with references to the trader and the PA itself.

Task agent contacts user

At any time after launch, the agent may contact the user via the PA. The agent sends an interactive document that may source events to be picked up by an agent script. The task agent might define a 'splash' screen to be sent to the user immediately after launch to gather additional configuration information.

User contacts agent

The PA maintains a list of active agents; agents the user has launched which have not yet terminated. The user may select any agent from this list to initiate a dialogue with the agent at its remote location. This kind of interaction is only possible if the agent defines a 'contact' operation which returns a document. The functions the user may select from this document are therefore application specific.

Agent reports to user

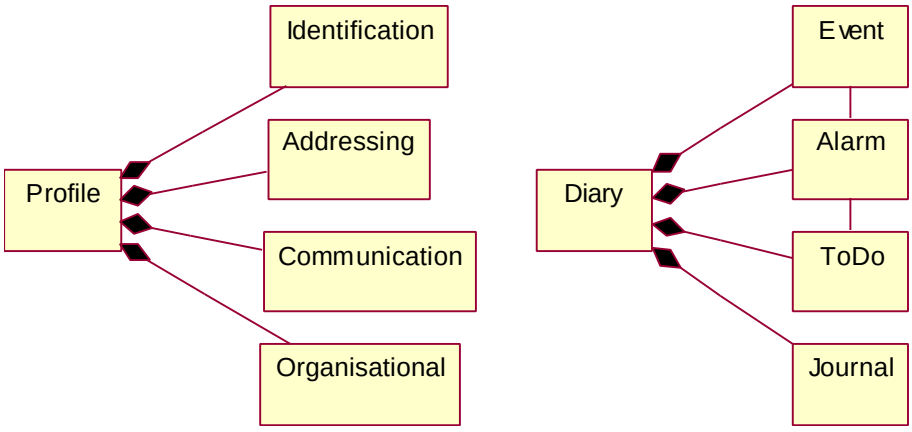
In many cases a fully interactive agent/user dialogue is not required. The agent simply needs to send a set of results back to the user. In these cases it is not necessary for the user to look at the results immediately. On receipt of the document the PA makes an attempt to locate the user by consulting the personal diary, searching for any current contact details. If the user cannot be contacted, the results are held in the information space where they may be read later.

Agent jumps to new location

The task agent may move to a new location by issuing a 'jump' command. The effect of this is to suspend the running of all mission components, including the script interpreter. Each component is responsible for saving its state ready for resumption of activity when it arrives. The jump thus hides the problems of thread suspension from the agent-programmer.

II. Personal Profiles

The function of this work-package is the storage and management of personal information. This information is very generic and may be used across many applications. It was decided early on to implement a personal profile and personal diary based on the vCard and vCalendar standards. It was necessary to convert these standards into an XML format so they could be manipulated with a standard set of tools. The profile and diary are containers for a number of specific types of data outlined in the domain model below.



Profile

A personal profile is a container for personal information. The profile may be loaded and saved in an XML format, but while in the information space the profile is implemented as a Java object. The design of the profile introduces a markup level design pattern in which we represent the same information in both structured and unstructured forms. For example, the user name is represented in an unstructured printable format, and additionally the same name may be broken down into its separate components such as title, forename, surname, etc.

Identification properties

The user may be identified by name.

Addressing properties

This represents the mail addresses at which the user may live or work.

Communication properties

This section aims to cover the major forms of electronic communication, such as telephone, fax, pager, email. The user may specify a number of communication forms for home and work.

Organisational properties

These properties reflect the kind of information found on your business card, including the name of the organisation you work for, and your role within that organisation.

Diary

The diary is a container for time-dependent information. The content of the diary may be loaded and saved in XML. The diary differs from the profile in that it is an active object able to source events which may be received by agents. The main technical challenge with the diary was to introduce mobility without 'dropping' any events that might occur in transit. In addition, the diary has to cope with moving between machines with clocks in different time zones, and which cannot (according to Einstein) be guaranteed to be 'synchronized' with respect to each other.

Alarm

An alarm denotes a particular point in time in the future, and may be set to repeat at regular intervals. Additional information about the reason for the alarm, may be provided by Event and ToDo elements.

Event

An event signifies a particular period of time during which the user may be occupied. For example, the user may be attending a meeting. The Event may be associated with an alarm which can be used as a reminder for the event. The event may also describe contact details which are valid for the duration of the event which can be used by an agent to get in touch with the user.

Journal

A journal entry is a time-stamped historical record. Agents may use journal entries to log their activity so the user can gain an idea of their progress in a task.

ToDo

A to-do entry describes an action to be performed at some point in the future. Used in conjunction with an alarm, the to-do entry provides additional information about why the alarm was raised.

Use-cases associated with the Personal Profiles work-package include the following:

Edit profile

We distinguish between a personal profile, and profile on the basis of usage. In itself a profile object may be used to store personal information. The term 'personal profile' designates a central profile object stored in the information

space. In normal use this personal profile is edited only by the user, however additional profile objects may be instantiated by task agents and accessed locally.

Edit diary

We distinguish between a personal diary, and diary on the basis of usage. The term ‘personal diary’ designates a central diary object stored in the information space. The user will use this diary to store information about their own whereabouts. Additional diary objects may be instantiated locally by task agents to schedule their own timed behaviour and to log their activity in journal entries.

Consult profile

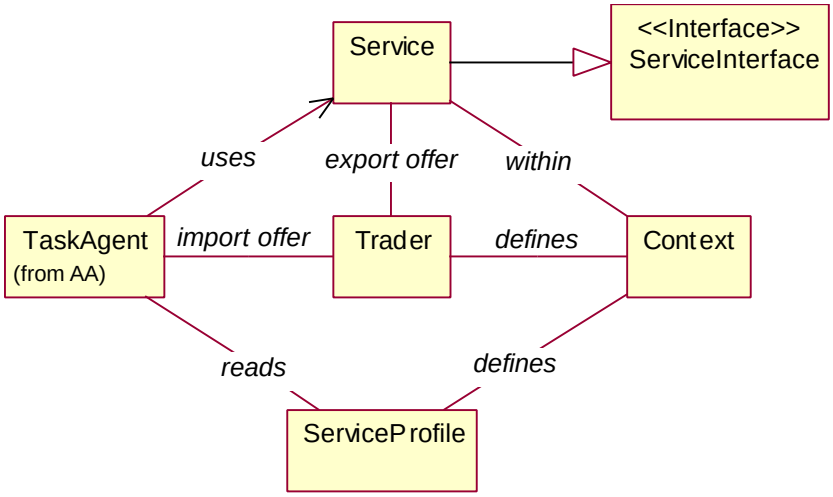
Task agents may consult the personal profile to gather information about the user they are working for. Agents performing this activity are likely to be working in conjunction with some service that has requested user details; for reasons of privacy, services are not able to access this information directly.

Consult diary

When a document is delivered to the Personal Assistant by a Task Agent, the PA may consult the personal diary to look for contact details associated with a given event. The consultation may be initiated by the diary itself in response to an alarm. For an alarm raised in the personal diary, the user may look for additional related information in the to-do list.

III. Service Interaction

This work-package provides tools allowing agents to locate and use services.



Trader

The trader provides a common, well-known service for locating other services. The trader acts as an implementation repository with which service offers may be registered. With appropriately defined context properties, the trader may also act as an interface repository, storing additional meta-data about the interfaces available within a context in the form of a service profile.

Service

The service itself is performed by a given service implementation, registered with the trader by a service provider. The service implementation must conform to the service interface associated with the context. The details of the service interface are reflected in the service profile.

Context

Service offers are organised within contexts. Each context defines a number of properties which may be displayed by each service offer within that context, and provide the basis on which a client may choose a given offer. For example, a property may associate a name with a given service. Personal Assistants are registered within a context that also

includes their user name, so that the user may locate their PA even after it has moved. Contextual properties may be updated by a service offer so as to reflect dynamic ‘quality of service’ features. Furthermore, contexts may be nested within each other, allowing services to be organised within a simple ontology.

Service profile

The service profile is an XML description of the service signature stored within the service context. The signature may be accessed dynamically by agents to perform type checking on structured XML data passed between the agent and service. The signature also supports the dynamic construction and invocation of operations on the service. However, the real value of the service signature lies in the fact that it documents the service interface in a way that is accessible on-line to agent programmers.

The principal Service Interaction use-cases are as follows. A new role is introduced for the Service Provider:

Export service Offer

The service provider registers the service implementation with the Trader, wrapping it up with a number of associated properties as a service offer. It is the responsibility of the service provider to maintain the service. While the service is available, the provider may update any dynamic service properties to reflect the current status of the service. When the service is no longer available, the offer should be explicitly removed from the Trader.

Import service offer

An agent, acting as a client of the service can request all the service offers within a given context. To be more selective, the agent is able to specify a number of constraints using a standard constraint language.

Interact with service

The task agent may interact with simple stateless service by invoking operations directly on the offer returned by the Trader. The Service Interaction work-package looks at a number of typical interaction scenarios; by a stateless service, we mean one where operations from any number of clients may be interleaved without error. More complex scenarios including two-way interactions may be constructed by defining additional interfaces within the service profile.