ESPRIT Project No. 25 338

Work package E

Personal Profiles

DE2: Requirements

ID:	DE2Requirements	Date:	19/12/97
Author(s):	Steve Battle	Status:	deliverable
Reviewer(s):		Distribution:	

Change History

Document Code	Change Description	Author	Date
DE1.1	Deliverable.	Steve Battle	19/12/97

PERSONAL PROFILES : Requirements document

Personal Profiles : Requirements document

Work package E relates specifically to the storage and maintenance of so-called *personal* profiles. It has become clear however, that profiles will be necessary to describe other components of the system. It is hoped that many of the techniques developed within this package will be available for re-use within these other areas.

Personal profiles are the means of storing long-term data that doesn't change too rapidly. All updates to the profile are initiated or sanctioned by user activity, so the rate of change is at least as slow at the human time frame. Any more frequent transactions are better suited to the *information space* of work package C. This requirement should ensure that in *most* cases the agent may work with simple copies of the profile, avoiding complex and expensive replication schemes.

Domain Object Model

We start with the development of the domain object model[Jacobson]. This will list all the main objects that have been identified in relation to personal profiles. At this stage, we will also include profile objects found in other parts of the system. This is partly to show that these objects have not simply been forgotten (and to prevent them from actually being forgotten), but will also mark out the system boundary between them and the personal profile. Each object is listed with a short description. In addition, the work package with ultimate responsibility for these `profiles' is listed alongside. The domain objects are shown in the following list. Where these objects are grouped together, or can be seen as aggregations of simpler objects, is indicated by indentation.

Personal profile	**WPE Personal Profiles**
user identification	*name, photograph?*
addressing properties	*for postal delivery & telecommunications*
security properties	*numeric PIN, public key, authorisation certificates*
placeMarks	*references to trustworthy followMe places*
spaceMarks	*references to objects in your information space*
serviceMarks	*references to favourite services*
agents	*active agents, agent parameters, personal assistant*
diary	*Calendar data, scheduled meetings/locations/people*
alarms & reminders	*`To do' list, recurrence rules*
service parameters	
portfolio	**WPI Pilot Application 1**
news select	**WPJ Pilot Application 2**
biographical	*lifestyle keywords*
Agent profile	**WPD Autonomous Agents**
goals	*Logical representation of agent behaviour*
scripts	*Expressed in agent description language*
constructor	*Agent initialisation*
main	*Procedural representation of agent behaviour (Mission)*
destructor	*Clean up after agent (particularly the information space)*
components	*Embedded Java objects*
Itinerary	*Representation of Agent mobility*
Service profile	**WPF Service Interaction**
service interface	*Interface data types and methods*
service form layout	*Abstract description of a user interface*
service contract	*Behavioural description in terms of the service model*
service model	*The entities and relationships within the service domain*
Device profile	**WPH User Access**
device capabilities	*Support for specific data types*
device driver	*Type and location (host) of specific drivers*
Place profile	**WPB Mobile Object Workbench**
place policies	*Security, Reliability*
parent directory	*Naming service*
Host profile	**WPB Mobile Object Workbench**
installed device drivers	**WPH User Access**

Many of these objects would be linked together. Examples of such linkages are:

- The personal profile refers to places and services.
- Your diary may refer to other people attending the same meeting.
- A diary alarm may refer to an agent that it will activate.
- An agent may refer to the user who created it.
- An agent may refer to the services it uses.
- An agent itinerary may refer to the places it visits.
- An agent may present a service interface.
- An agent may carry its own diary for timed behaviour and scheduling of delivery times and locations.
- A service may refer to example agents.
- A place would refer to the host on which it resides.

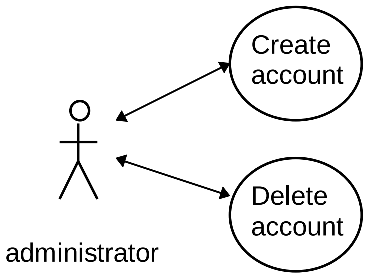
Finally, a number of miscellaneous properties have been identified (WP_I_Processes V. 1.0) which, because of their dynamic nature, seem to belong in the information space, and not in the personal profile.

- Agent log
- User online address
- Service `cookies’

Use Case Model

Use cases are a way of specifying how a system is actually used. They provide a high level description of system functionality without prescribing a specific class structure. Each *use case* bubble denotes a sequence of actions initiated by an *actor*, which represents a particular role that may be instantiated by a real person or even another system.

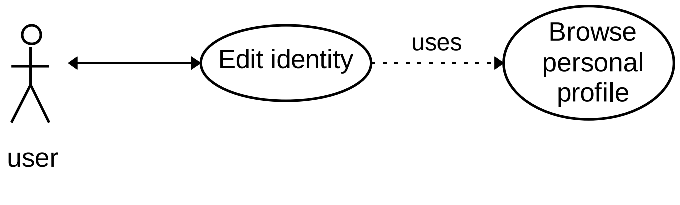
The requirements captured in these use cases are partly based on the internal project document, “Process descriptions for pilot applications” (Hans-Guenter Stein), and also on discussions held at followMe technical meetings. This document deals only with use cases that have a direct impact on Personal Profiles. In the first use case diagram we separate administration of the user account from its everyday use. The use of the administrator role does not imply central administration; the administrator and user may be one and the same.



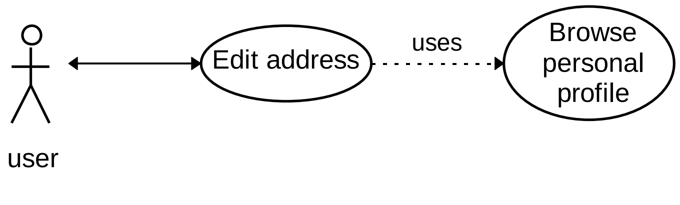
1) Create account: To create a new user account, a new Personal Assistant is started. The constructor for the Personal Assistant creates a new personal profile with the appropriate user name and default PIN.
[source: Process descriptions for pilot applications, create account; Initial authentication and PUA registration]

2) Delete account: To delete an account, the administrator role destroys the Personal Assistant. The PA destructor specifies procedures for terminating active agents and cleaning up objects in the information space. The personal profile is withdrawn.
[source: Process descriptions for pilot applications, Retire from FollowMe]

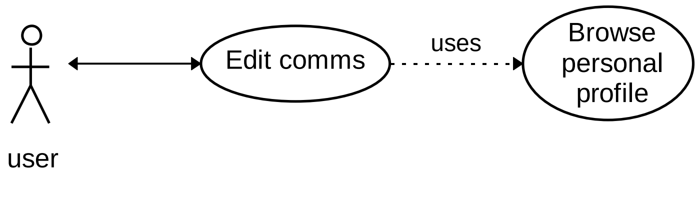
The following use cases can all be seen as functions of the personal assistant, and in particular the functions that the personal assistant performs in relation to the personal profile. We can extract a new use case which represents this commonality, *browse personal profile*. The connection between *concrete* use cases and this *abstract* use case is shown by the *uses* relationship.



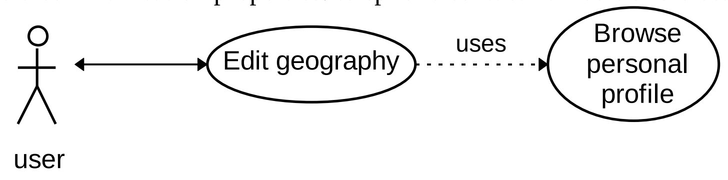
3) Edit identity: Change user name and globally unique identifier.



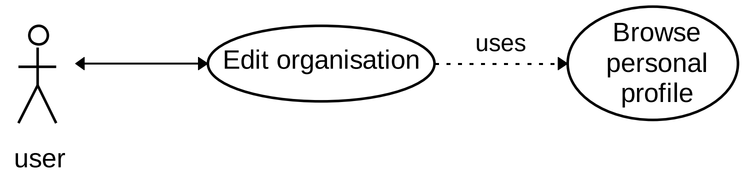
4) Edit address: Change postal address for delivery services.



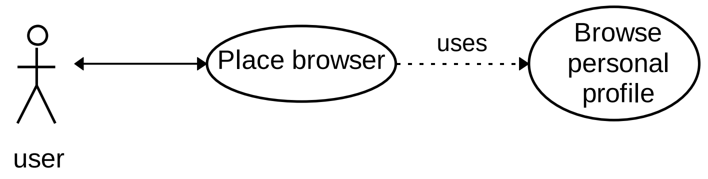
5) Edit comms: Change tele-communication properties: telephone contact numbers and electronic mail.



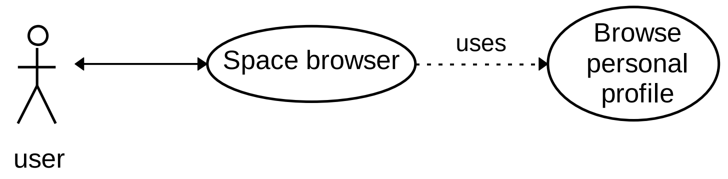
6) Edit geography: Change properties related to the geography: country code.



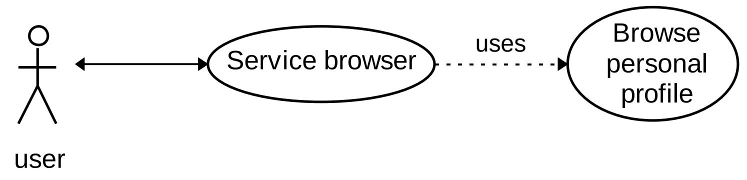
7) Edit organisation: Change organisational properties: Job title, organisation name.



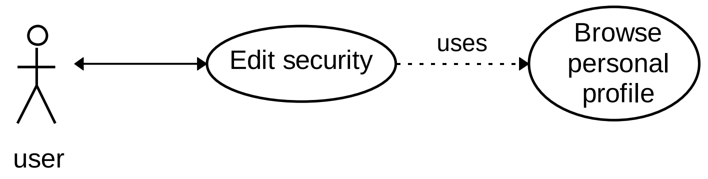
8) Place browser: Maintain list of known places and their profiles.



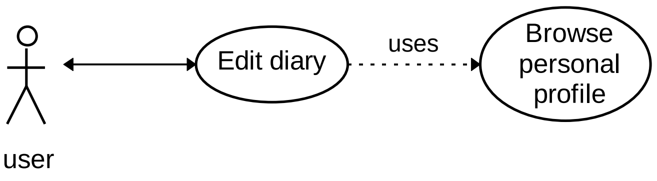
9) Space browser: Maintain list of information space entries, which may contain results delivered by agents.



10) Service browser: Maintain list of known services, including links to agents designed to operate with them. The user is able to start new agents and check the status of currently running agents.
[source: Process descriptions for pilot applications, new service; retire service; create_new_TA]

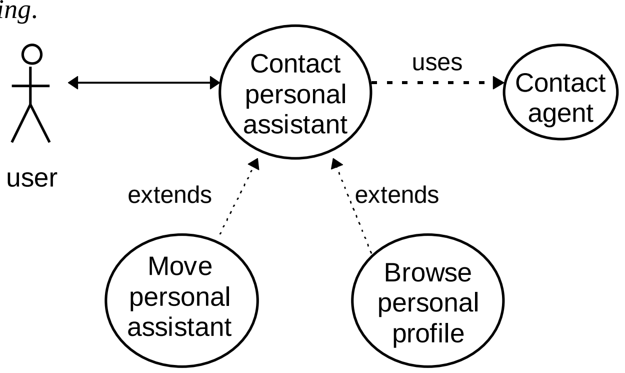


11) Edit security: Change PIN.
[source: Process descriptions for pilot applications, change password]



12) Edit diary: The user is able to create/update/delete scheduled diary events and reminders. In addition to being general resource for the user, the diary may be tailored for specific agent behaviour.
[source: Process descriptions for pilot applications, manage diary]

While all use cases must specify a complete behaviour in their own right, they may be extended to include new behaviour. For example, we may want to add new exception handlers or additional user options. This *extends* relationship is comparable to *subclassing*.



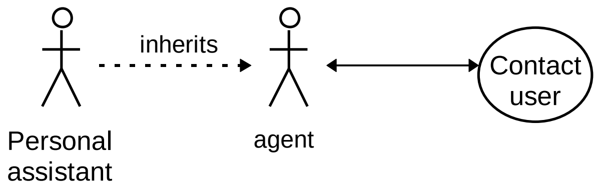
13) Contact personal assistant: Browsing the profile is just one of the operations the user can carry out with the personal assistant. In addition to this the user has control over its movements between followMe places. These are seen as behaviours which are added to *contact personal assistant.*
[source: Process descriptions for pilot applications , Link to PUA; PUA main menu]

14) Browse personal profile: As described in the architecture Report (E), each profile object may correspond to an abstract view. These views may be presented to the user via the magic of the personal assistant.

15) Move personal assistant: This provides an interface to the functions provided by the Mobile Object Workbench API in relation to the personal assistant. Invoking *moveTo* (Architecture Report, B) allows the user to move the personal assistant to a new place.
[source: Process descriptions for pilot applications , move to new location]

16) Contact Agent: Contacting the personal assistant is seen as a special case of contacting an agent. To contact an agent, the user must first identify themselves by entering their PIN. Once verified against the personal profile, the agent presents an interface to the user.
[source: Process descriptions for pilot applications , User requests system access

Use cases may also capture the system from the point of view of the personal assistant which can be seen as just another actor. The notion that the personal assistant is a special case of agents in general is shown with the *inherits* relationship between roles.



17) Contact user: An agent can contact a user as long they know where to reach them. If the user is successfully contacted, the agent may present them with an appropriate interface.
[source: Process descriptions for pilot applications , request immediate report; schedule report; TA_contact_user]

References

Mike Bursell, Douglas Donaldson et al (APM), *Architecture Report DA1.1*, DA1.1 V.1.0
Hans-Guenter Stein (FAST), *Process descriptions for pilot applications*, WP_I_Processes V. 1.0
Ivar Jacobson, *Object-Oriented Software Engineering: A use-case driven approach*, Addison-Wesley, 1992.
Donald G. Firesmith, *Use Cases: the Pros and Cons*, http://www.ksccary.com/usecjrnl.htm