FollowMe Agent Framework

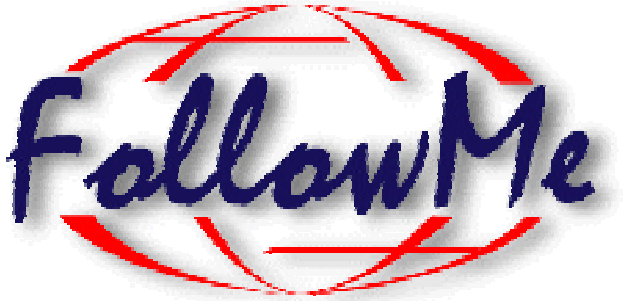
Michael Yearworth

Intelligent Computer Systems Centre

University of the West of England, Bristol

my@ics.uwe.ac.uk

ANSA Technical Board 21/7/98



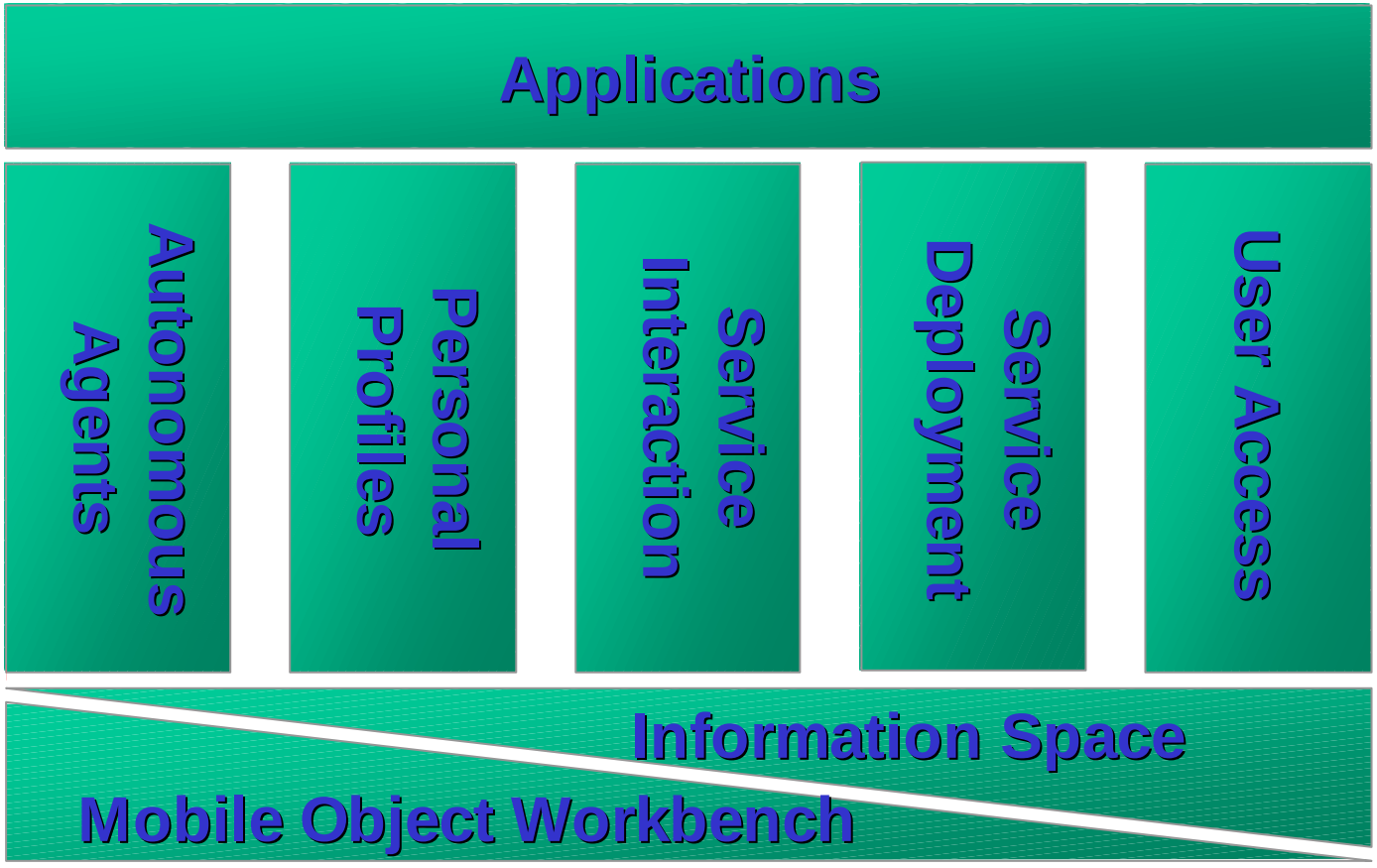
Freeing the User from the Fixed Desktop

APM (UK), FAST(DE), INRIA(FR), TCM(FR), UWE(UK)

FollowMe Objectives

- Creation of an infrastructure to support mobile working
  - ⇨Agents *as well as users* are mobile
  - ⇨Multiple locations - home, work, on-route
  - ⇨Multiple devices - PC, phone, FAX
    - ⇨Java enabled devices
- Using mobile object technology
  - ⇨Flexinet, MOW from ANSA

FollowMe Architecture



FollowMe Agent Framework

- Agent scripting mechanism integrated with support for agent and user mobility
  - ⇨what to do, and where, and how
- Personal Profile
  - ⇨personal details of owner including diary and locations
- Service Interaction
  - ⇨agent interactions with the environment

Agent Definition

- Jennings & Wooldridge definition of agents (weak agency model):
  - ⇨Autonomous
  - ⇨Social
  - ⇨Responsive
  - ⇨Pro-active

Autonomy

- Autonomous Behaviour
  - ⇨Implemented by design patterns - Command, Strategy, Delegation (GoF)
  - ⇨Mobility is another means of achieving autonomy and is provided by the MOW
  - ⇨Autonomy is also achieved by a Connection object which decouples agent *and* user mobility

Sociality

- Social Behaviour
  - ⇨CDP Agent limited to this definition
  - ⇨Agent Interaction with
    - ⇨*users*
    - ⇨*services*
    - ⇨*other agents*
  - ⇨FollowMe addresses all three cases
  - ⇨Interoperability is managed by service and mission profiles

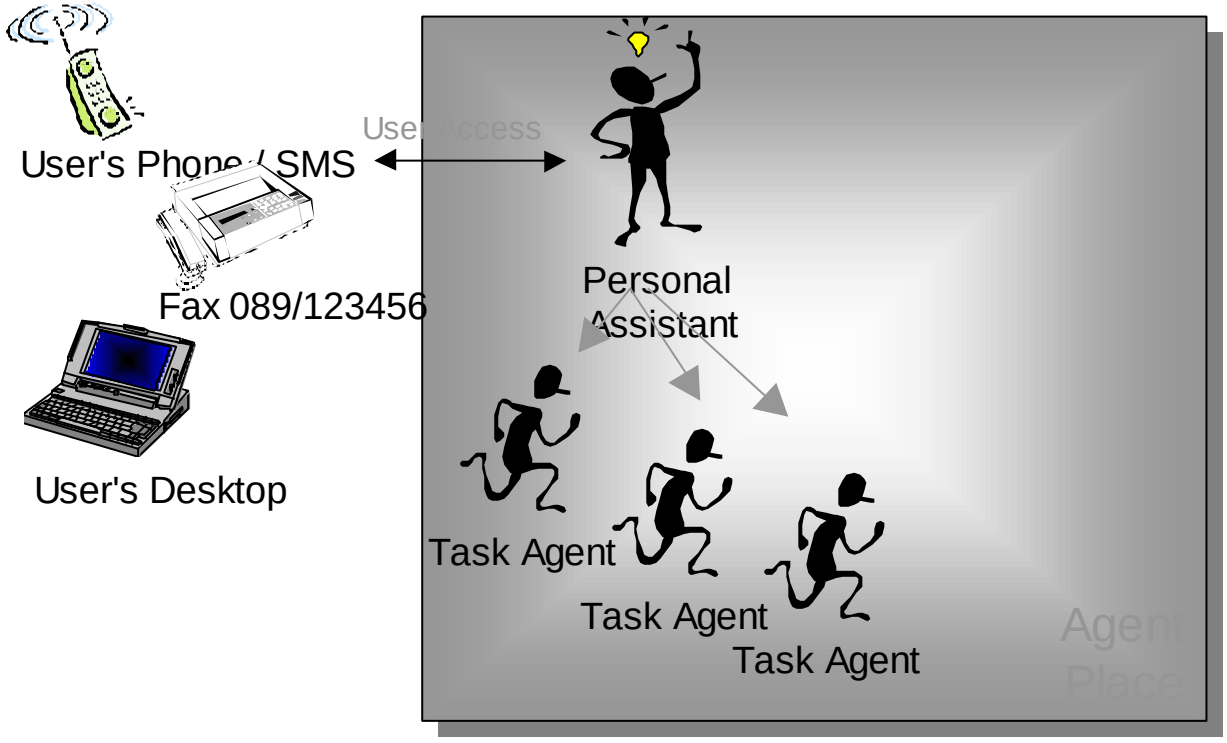
Responsiveness

- Responsive Behaviour is mainly expressed by mobility under different conditions
  - ⇨If environment threatens continued existence e.g. if a place will cease to exist
  - ⇨If communication can only occur in a trusted place
  - ⇨As an optimisation strategy e.g. with an information filtering agent

Pro-Activeness

- Goal oriented or *intelligent* agent behaviour
  - ⇨FollowMe architecture capable of supporting agents with as much, or as little, intelligence as is needed to complete a task
  - ⇨Hooks exist in service interaction design for supporting selection of agent mission on basis of behaviour

Agent Framework - Autonomous Agents



Agent Scripting

- Scripting approach chosen because…
  - ⇨We are connecting components together
    - ⇨*Component glue*
  - ⇨Decouples users from implementation details of the MOW and IS
    - ⇨*Easy use of mobility, profiles and user access*
  - ⇨Strong link with the desktop
    - ⇨*ECMAScript*
  - ⇨Easy language for prototyping
    - ⇨*Agent missions*

Autonomous Agents Components

- Personal Assistant
  - ⇨Focus of interaction with user
  - ⇨Facilitates selection, initialization and launching of task agents
  - ⇨Manages user's personal profile and diary
  - ⇨Mediates all communication between user and agents

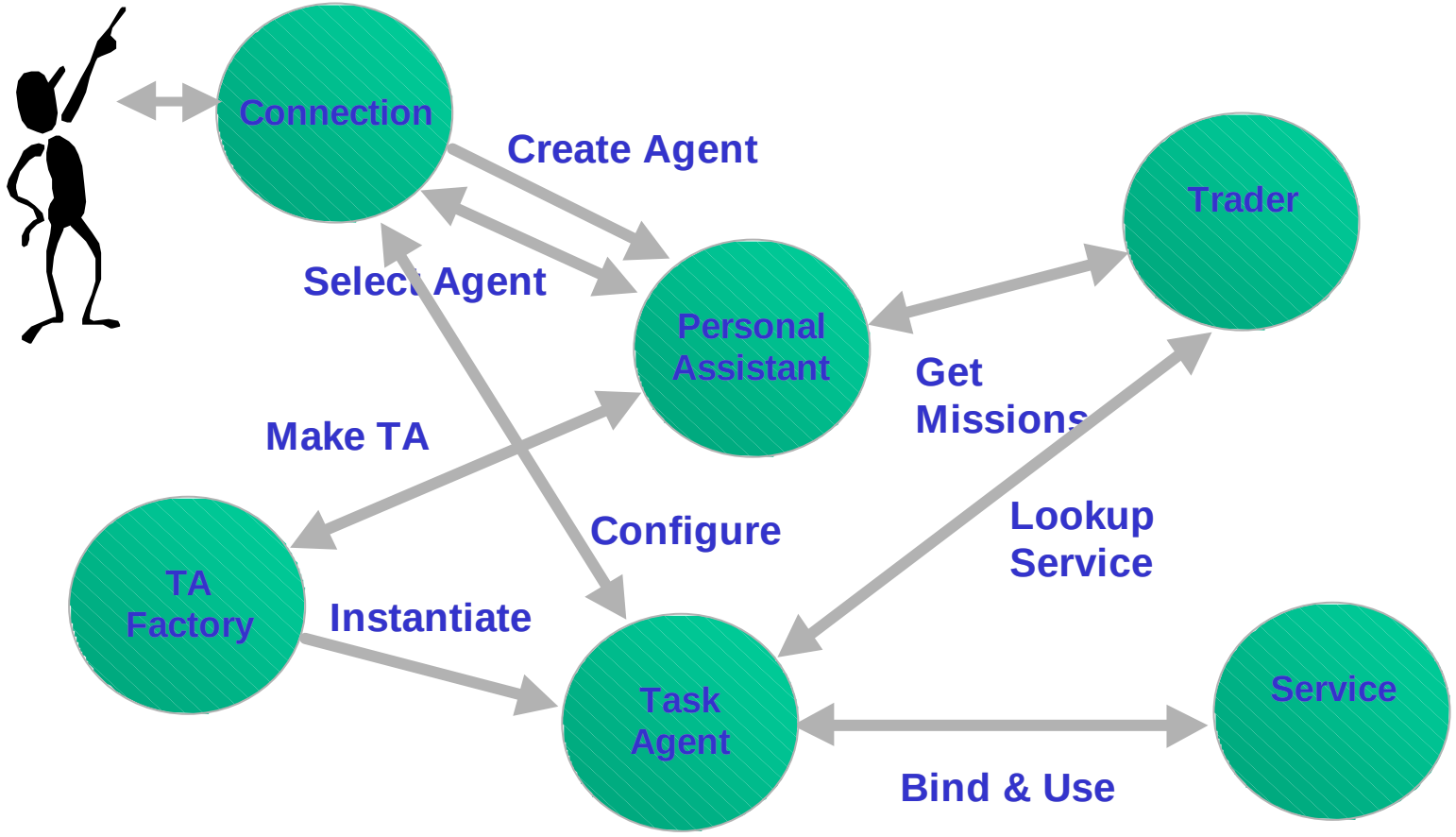
Agent Missions

- Personal Assistant enables selection of agent missions
  - ⇨Extended Trader holds agent missions
  - ⇨Personal Assistant enables user to browse and select missions
  - ⇨Mission script is then passed to Task Agent Factory to create Task Agent
  - ⇨Agent mission can then be customized

Autonomous Agents Components

- Task Agent
  - ⇨Contains the agent mission expressed as a script and created by Task Agent Factory
  - ⇨Task Agent runs script, binding to Services and moving to Agent Places as required
  - ⇨TaskAgent communicates with user as required via Connection object

Agent Framework - summary

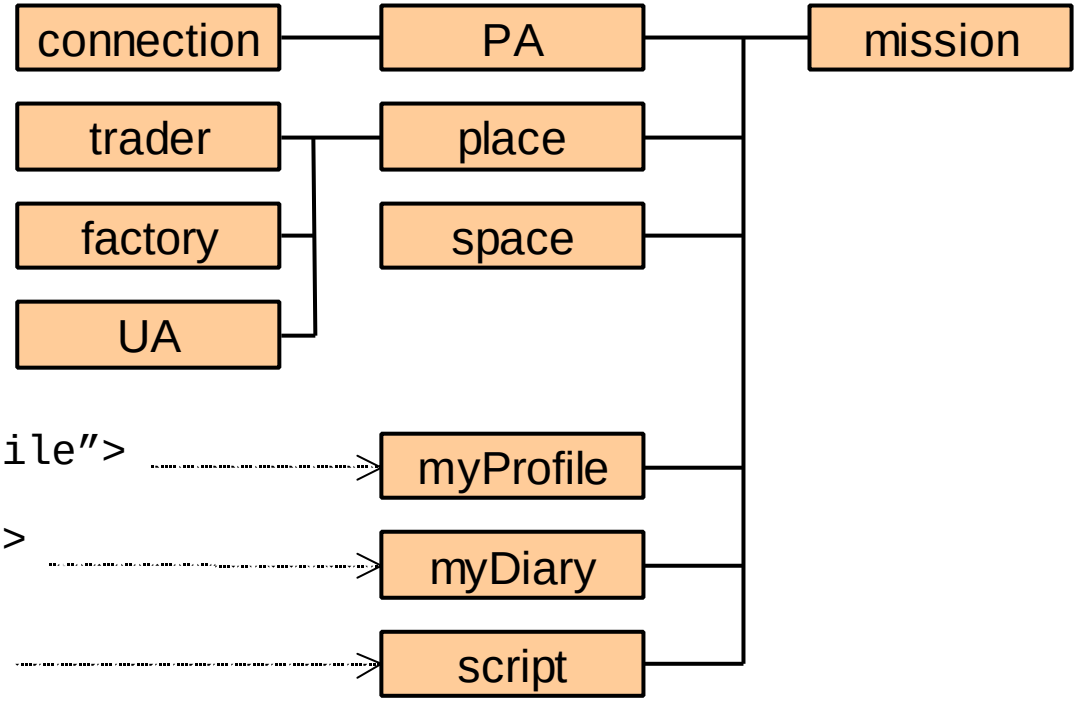


Creating and Using Agents

- Scripts written in ECMAScript
- Embedded in XML documents
  - ⇨`<SCRIPT> </SCRIPT>` *tags*
- Component hierarchy
  - ⇨*Mission is the parameterization of an agent (c.f. strategy & command design pattern)*
- Access to built-in and 3rd party objects

Mission with Parameters

Built-in objects:




```
<MISSION>
  <PROFILE name="myProfile">
  </PROFILE>
  <DIARY name="myDiary">
  </DIARY>
  <SCRIPT> … </SCRIPT>
</MISSION>
```

Access to Mobility from Script

- Interface to low level support in MOW
- Requires shut down of active threads
- Complexity of managing threads hidden by the script interpreter
- Programmer is free to include jumps in code blocks like loops and functions

```
<SCRIPT>
  for (i=0 ; i<itinerary.length ; i++) { jump(itinerary[i]) ; ... }
</SCRIPT>
```

Script Interpreter Design

- Explicitly implemented execution stack object
  - ⇨Script evaluated using this execution stack
  - ⇨On encountering a jump(), execution stack and variables serialised
  - ⇨Execution stack and variable objects move to another place
  - ⇨Script interpreter resumes operation

Simple Example Script

- Send agent to buy a pizza

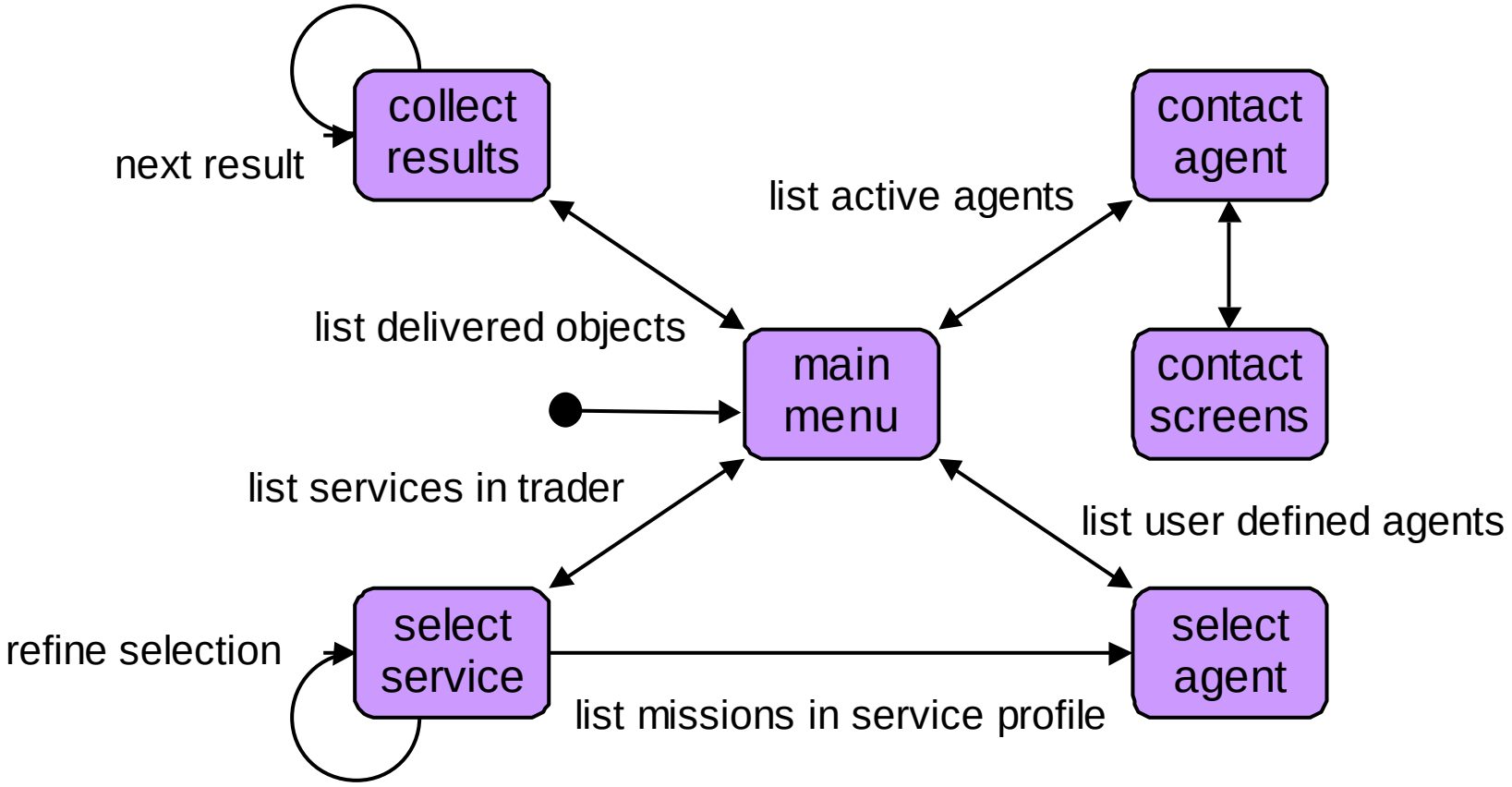
```
<mission name = " PizzaFactory" >
  … forms defined here
   function ok() {
     jump(service) ;
     order = service.orderPizza(base.value) ;
     for (i=0 ; i<topping.length ; i++)
       if (topping[i].checked) order.extra(topping[i].value) ;
     pizza = order.finished() ;
     send("pizza") ;
   }
service = place.trader.resolve(" PizzaFactory") ;
   if (service!=null) send(PizzaMenu) ;
</script>
</mission>
```

Example continued...

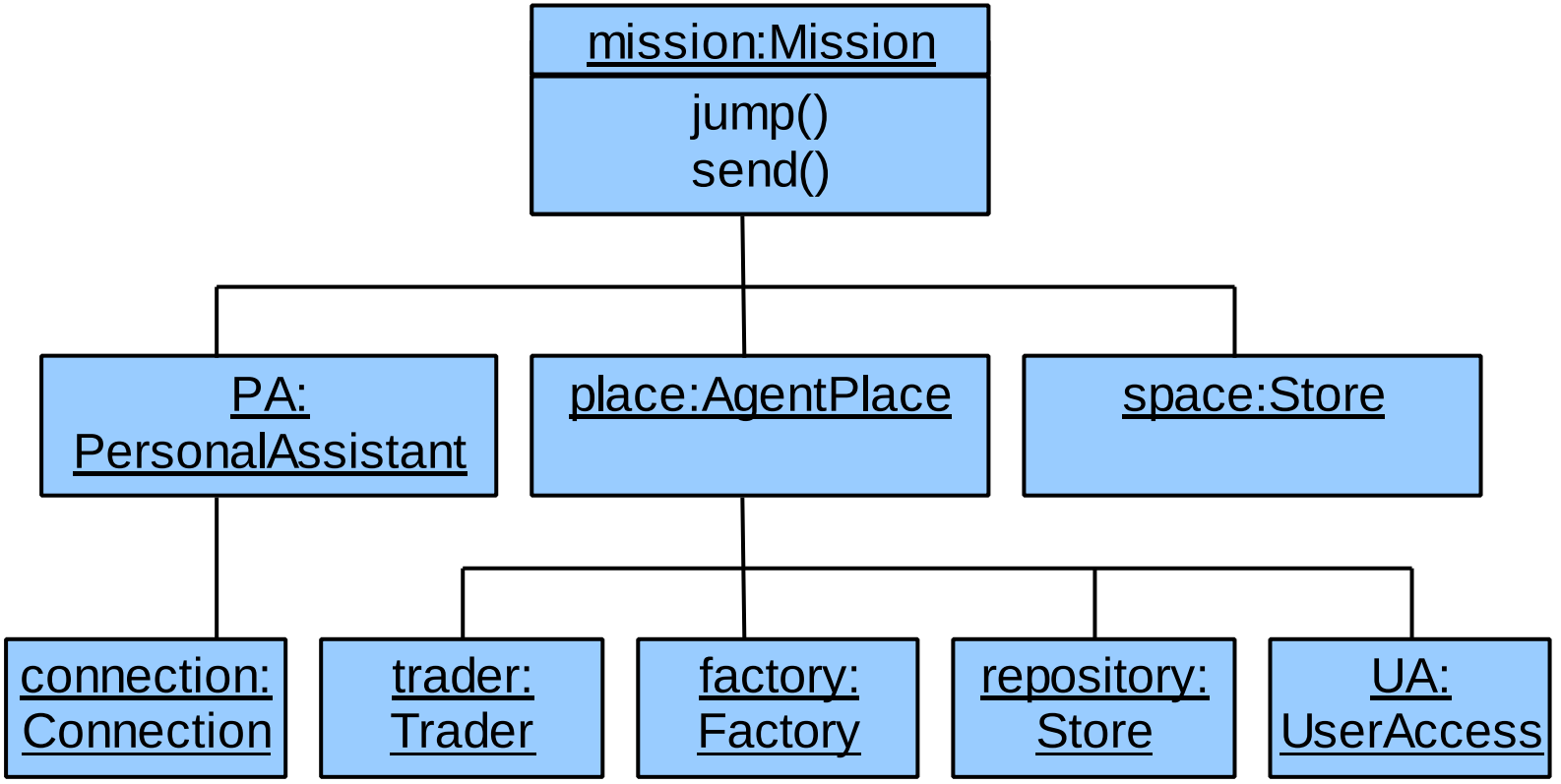
- Forms for user access

```
<form name = " pizzaMenu">
    Pizza base :
    <select name = "base">
      <option>cheese & tomato</option>
      <option>vegetarian</option>
      <option>ham & mushroom</option>
    </select>
Extra toppings :
    Anchovies <input name="topping" type="checkbox" value=" anchovies"/>
    Olives    <input name="topping" type=" checkbox"  value="olives"/>
    Cheese    <input name="topping"  type=" checkbox"  value="cheese" />
    <input type="submit" name="OK" value="OK"  onClick="ok()"/>
    <input type="submit" name=" cancel"  value=" cancel" />
  </form>
```

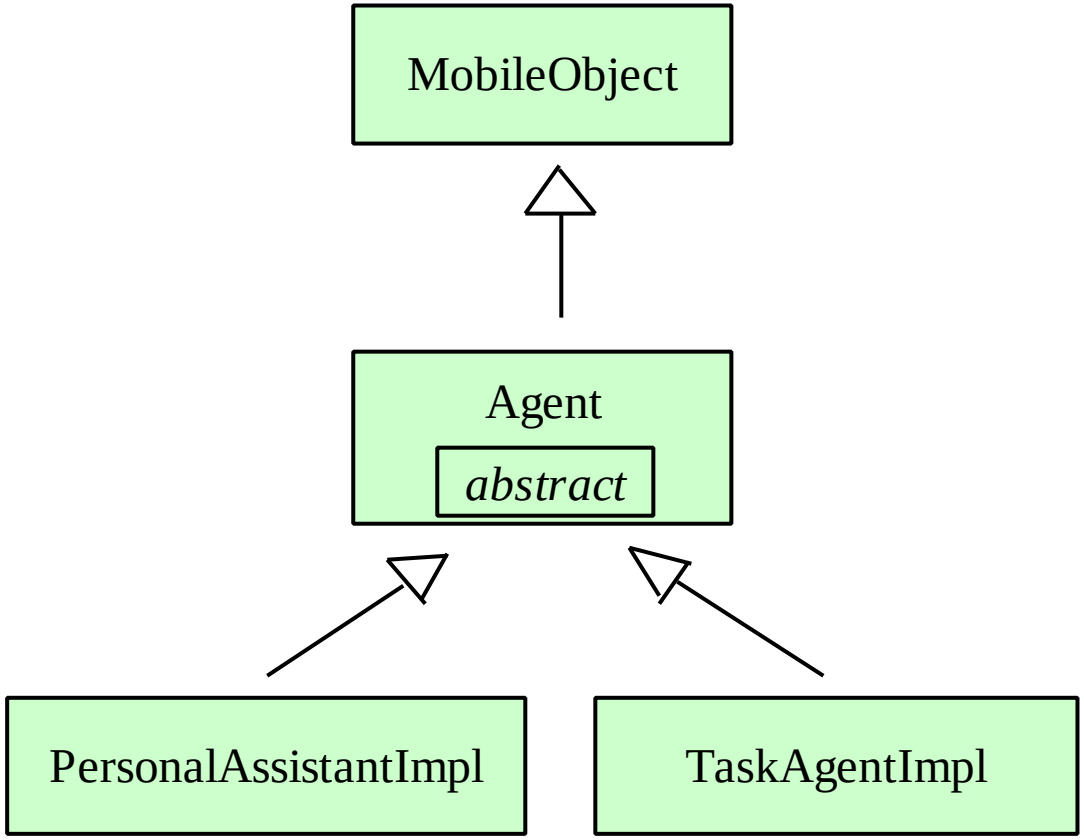
PA State Diagram



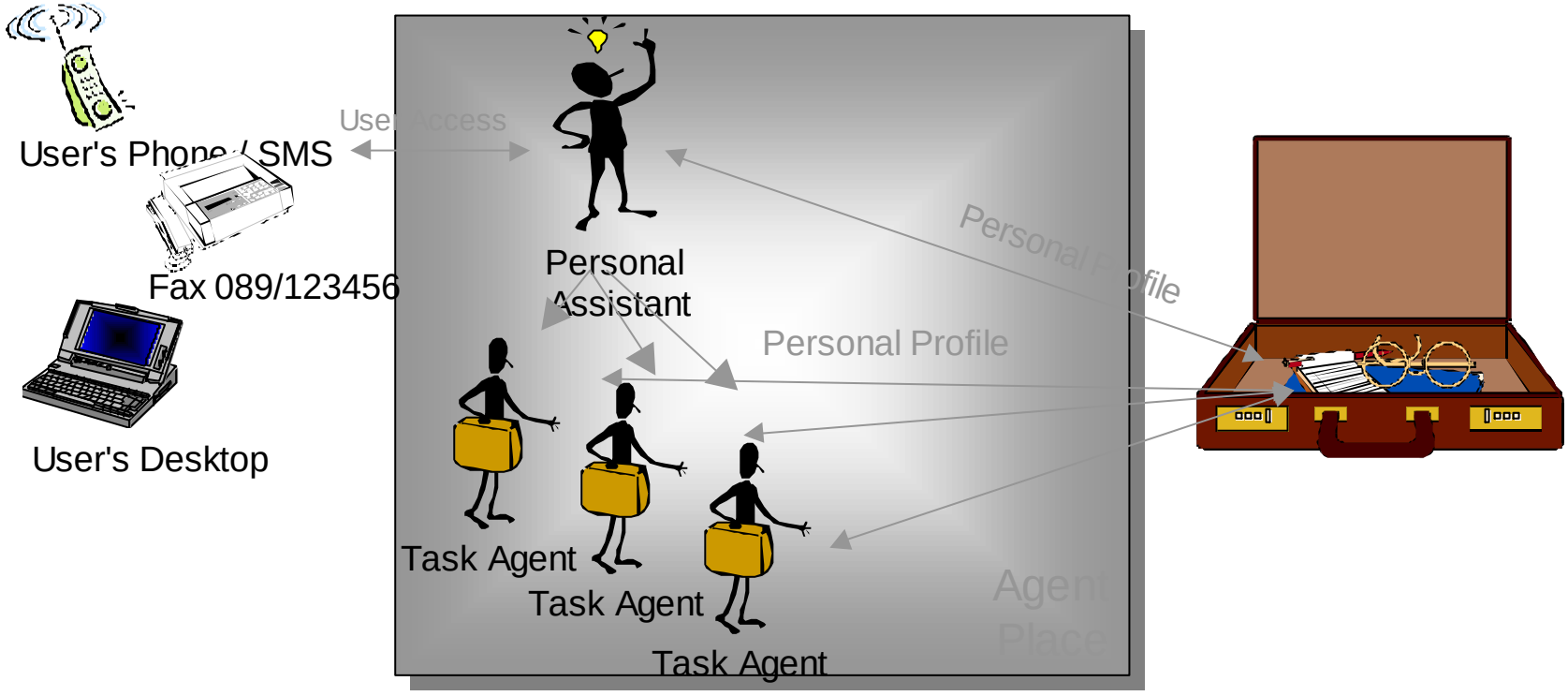
Object Hierarchy



Relation to MOW

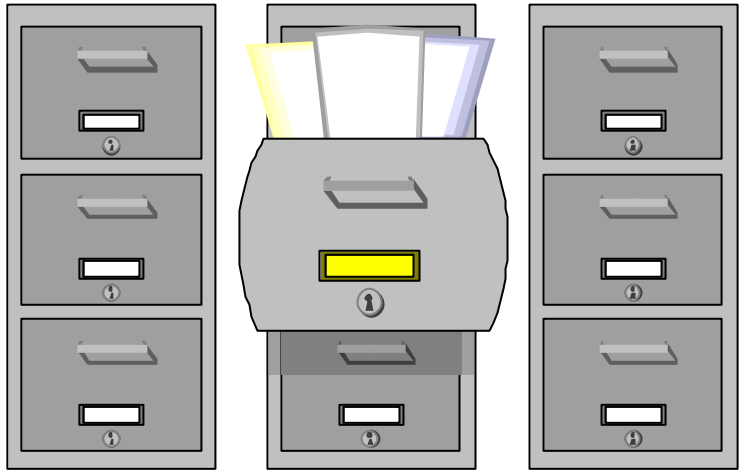


Agent Framework - Personal Profiles

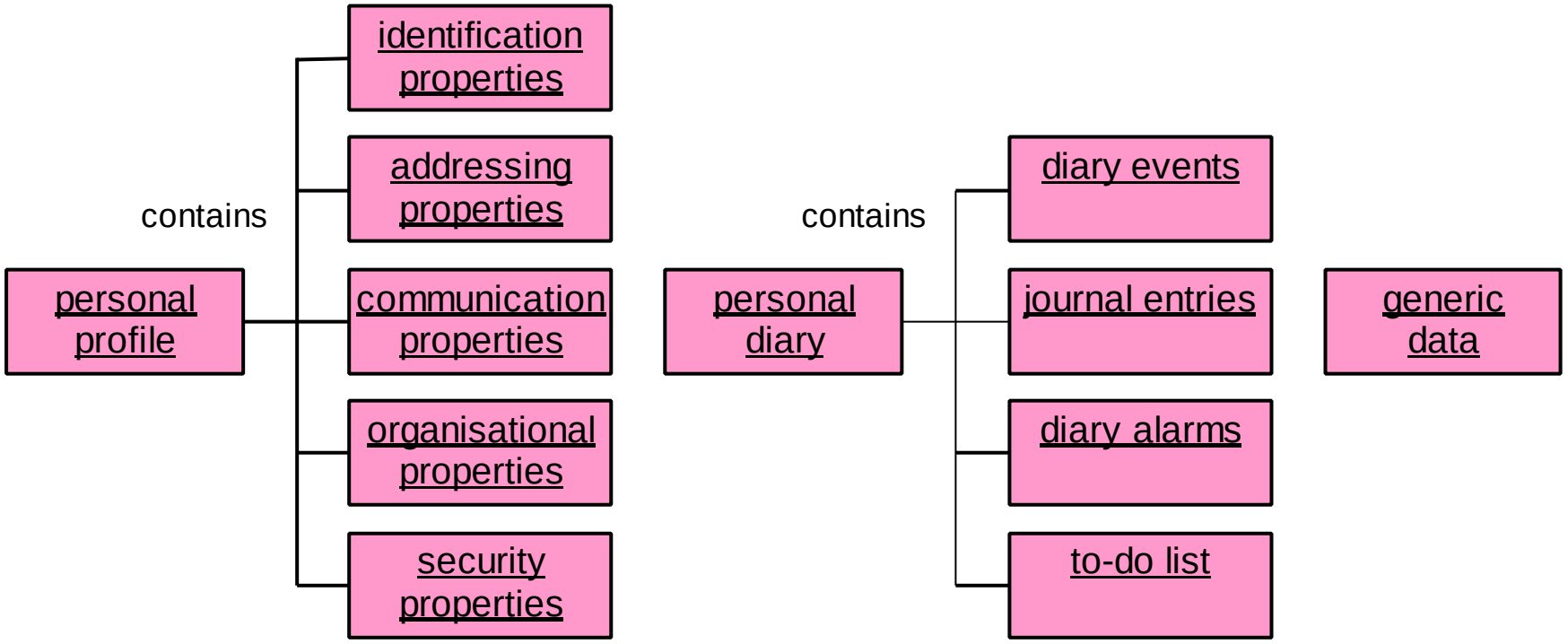


Personal Profiles

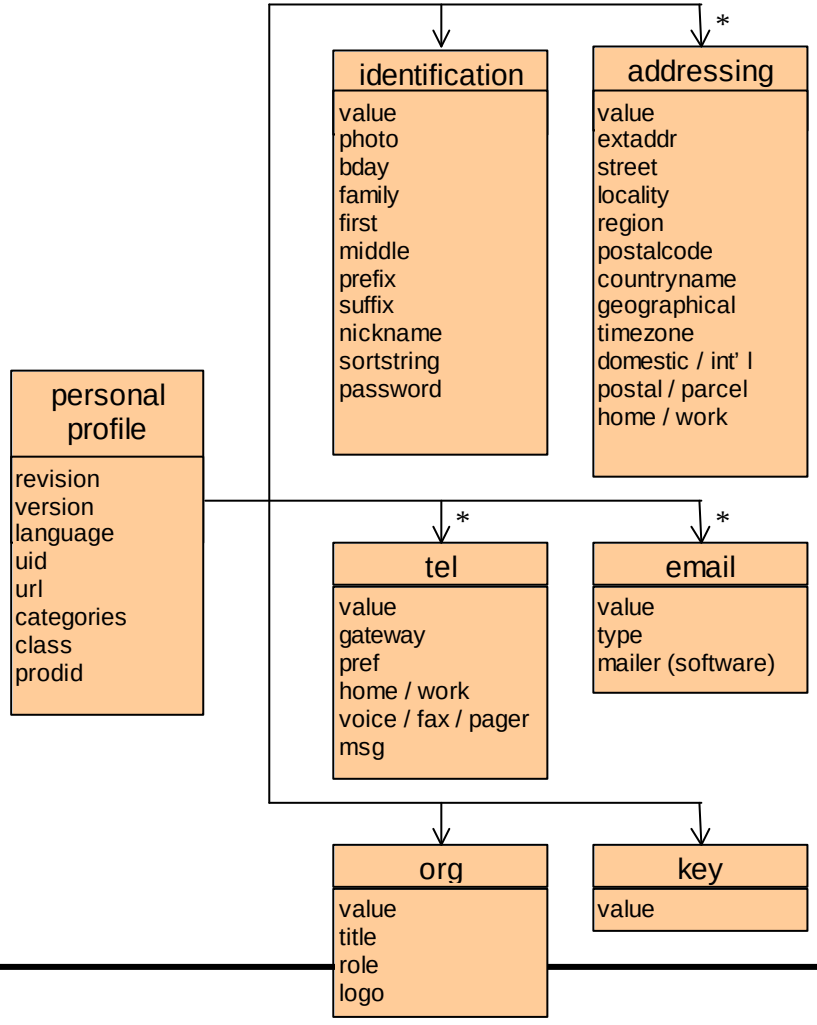
- Profile is a repository for personal information
- Available to mission script
- Created by user
  - ⇨Personal information
  - ⇨Diary information
    - ⇨*location data*



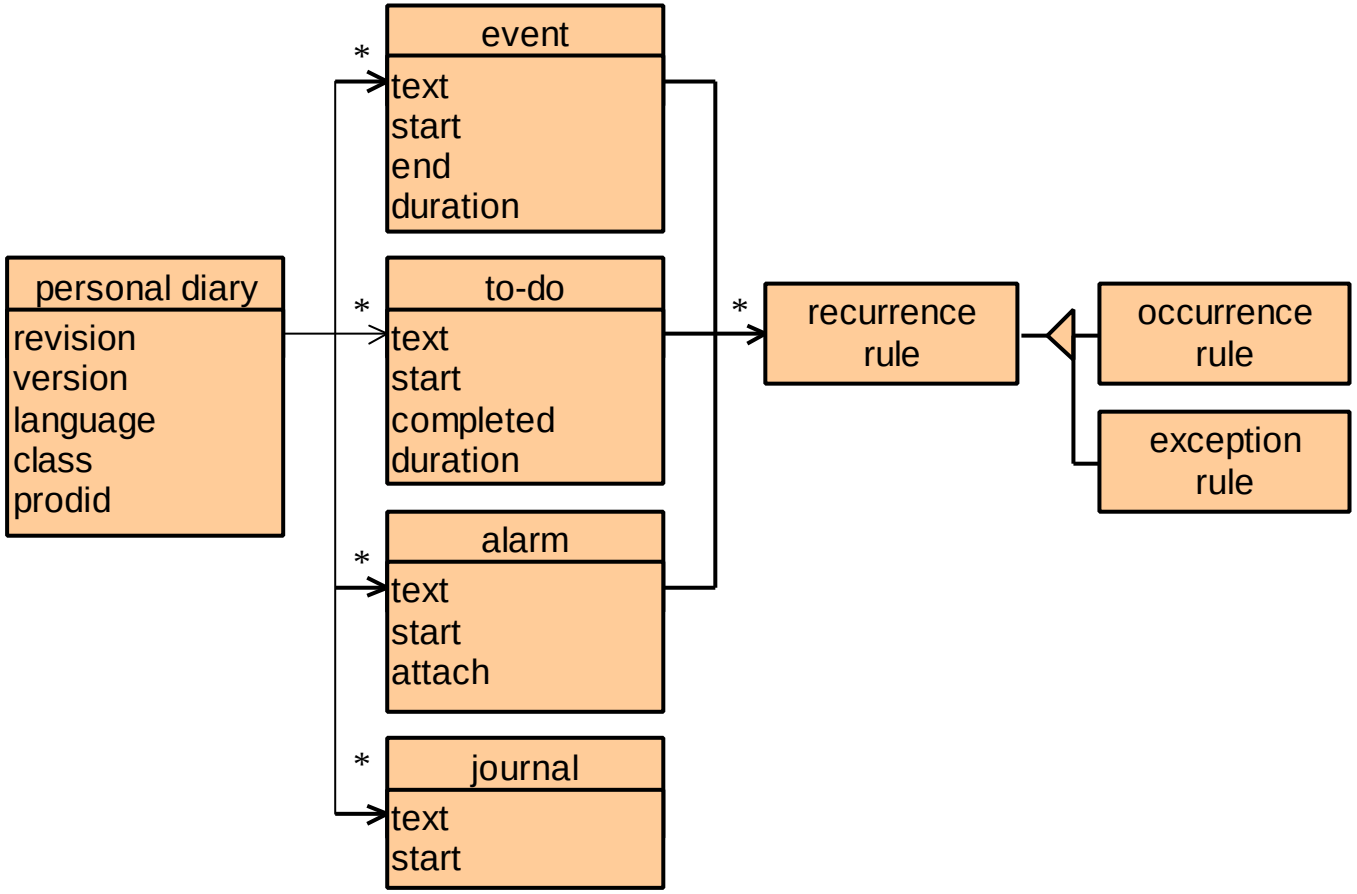
Personal Information



Personal Profile Objects



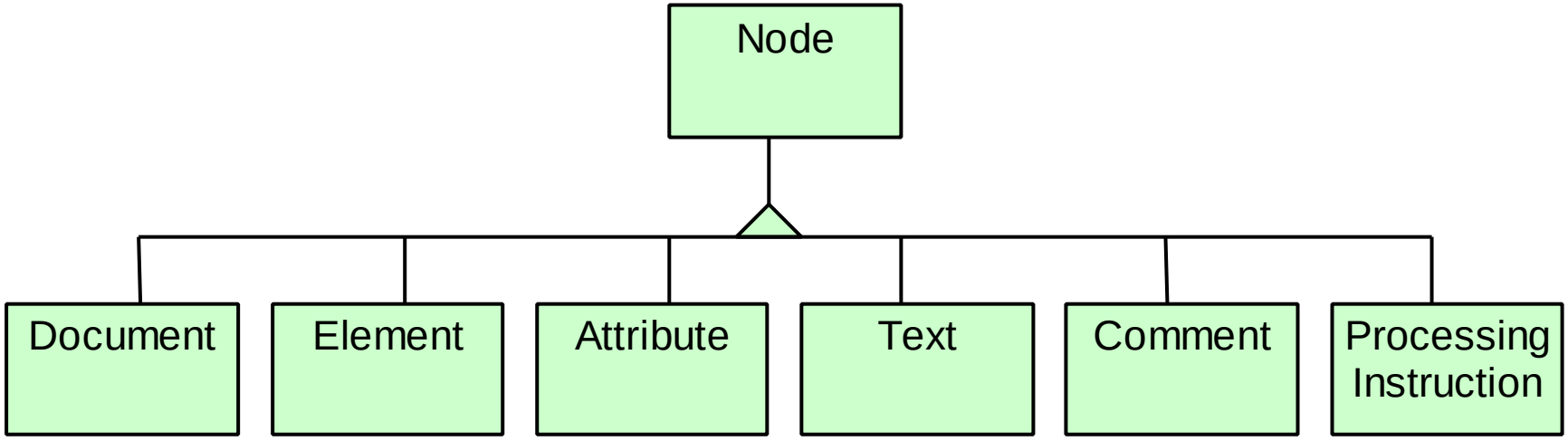
Diary Objects



Profile Repository

- Personal Profiles implemented using the Document Object Model (DOM)
- DOM hidden from mission scripts by profile API
- DOM API available for storing and querying other types of profile
- When Profile objects are externalised, XML is used as representation

Document Object Model



Profile Representations

- Machine and/or human readable profile

```
<PROFILE>
  <ID>Steve Battle</ID>
</PROFILE>

<PROFILE>
  <ID first="Steve" family="Battle" />
</PROFILE>

<PROFILE>
  <ID first="Steve" family="Battle">
    Steve Battle
  </ID></PROFILE>
```

Pre-Defined Diary Tags

- Alarm

```
<DIARY>
    <ALARM START="19630606" ATTACH= "send(birthdayCard)"/>
      <ORULE FREQ="Y"/>
      </ALARM>
</DIARY>
```

- Journal Entry

```
<DIARY NAME= “Captain’s log”>
      <JOURNAL START= “22591231”>
      Exchanged smalltalk with the Vorlon ambassador today.
      </JOURNAL>
</DIARY>
```

Pre-Defined Diary Tags

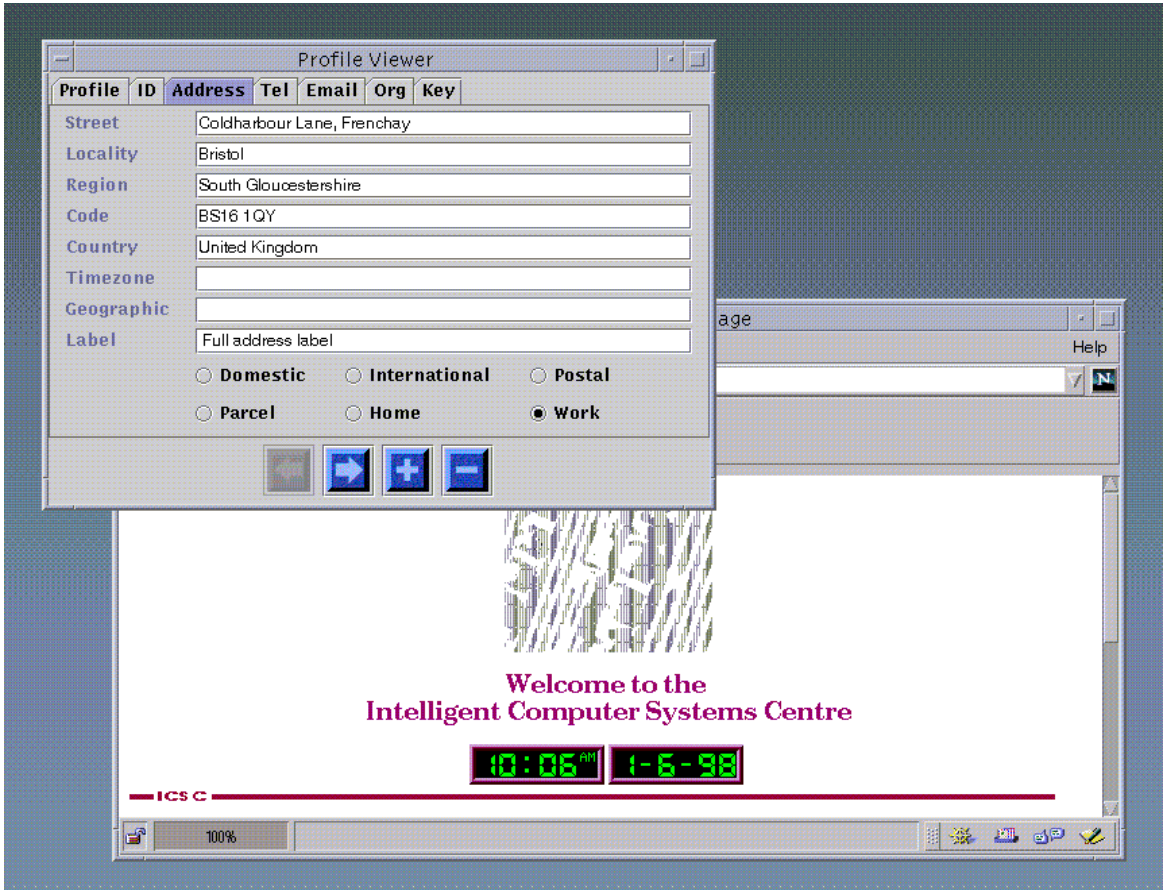
- Events

```
<DIARY>
  <EVENT START= “19980101T0900” DURATION= “PT8H”>
    Just another day in the computer business.
   <ORULE FREQ= “D:MO,TU,WE,TH,FR”/>
  </EVENT>  </DIARY>
```

- To Do lists

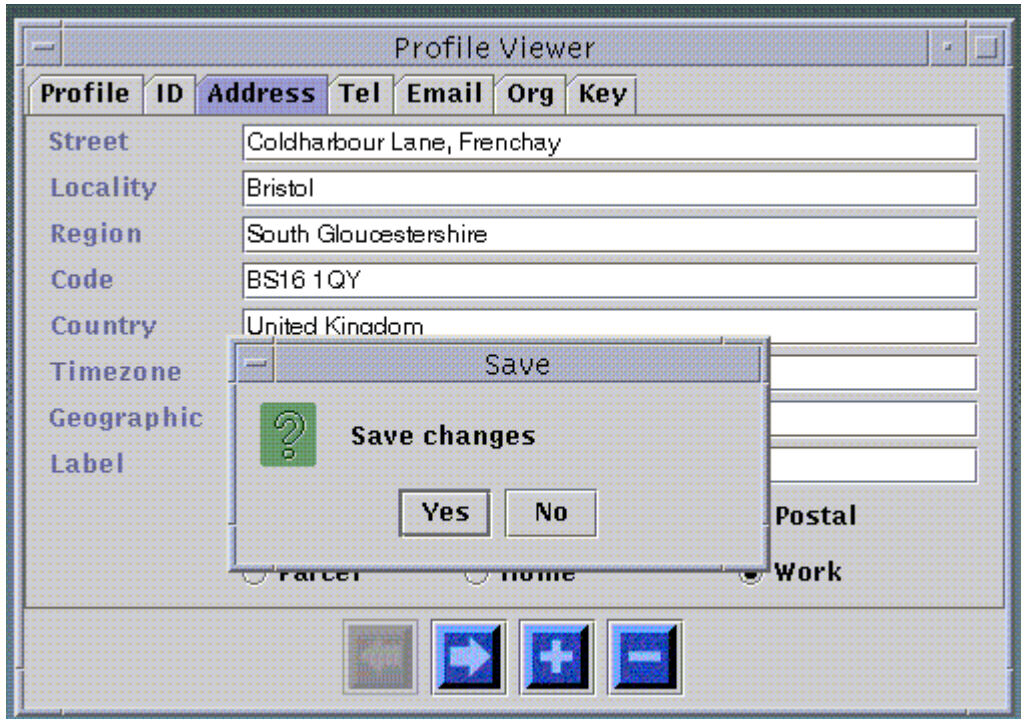
```
<DIARY>
 <TODO>
  <ORULE FREQ=”W’ MODIFIER=”MO”>
    Don’t forget the level 0 dump of system after 5PM today
 </TODO> </DIARY>
```

Profile Editor Implementation



Package

Profile Example



Package

Extending the Example

- Returning the user's name

```
// Extract user's full formatted name
userName = personalProfile.get_id().get_fn();      // formatted name
```

- Finding list of user's email addresses

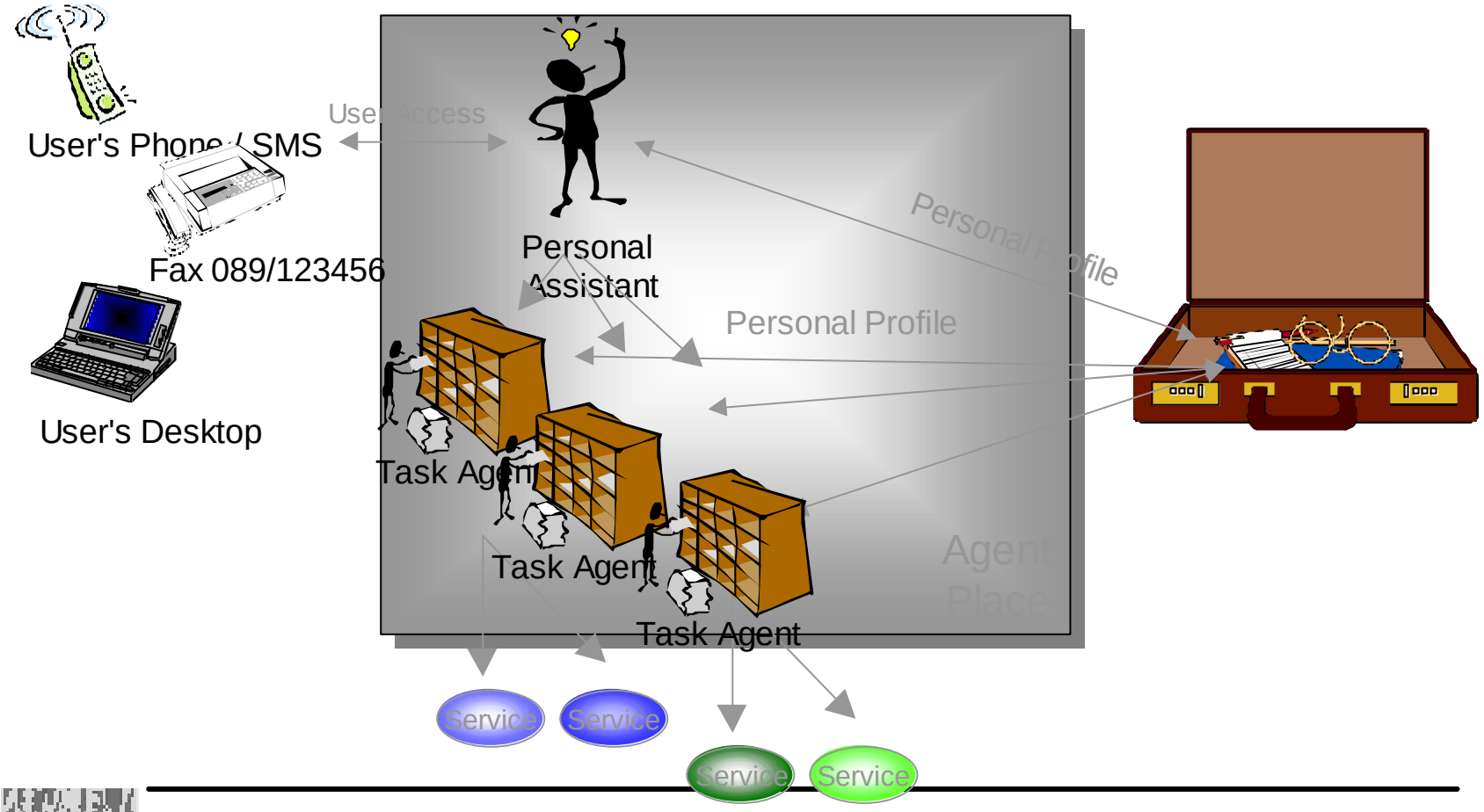
```
// display (using write function) all email addresses associated with user
emailEntries = personalProfile.length_email();     // number of Email entries
while (count < emailEntries) {
    thisEmail = personalProfile.get_email(count);  // current Email details
    write(thisEmail.get_text());                   // email address
    count++;
}
```

Further Example of Profile

- Find the user's home address label

```
// Find home address label if it exists
adrEntries = personalProfile.length_adr();    // number of Address entries
for (i=0;i<adrEntries;i++) {
    thisAdr = personalProfile.get_adr(i);       // current Address details
    isHomeAddress = thisAdr.get_home();    // boolean home status of details
    if (isHomeAddress) {
        homeLabel = thisAdr.get_label();
    }
}
```

Agent Framework - Service Interaction



Service Interaction

- Distinguish between basic service interaction and meta-level service interaction
  - ⇨Basic level provides for a working service interaction model for FollowMe
  - ⇨Meta-level interaction is a research oriented topic looking at behavioural semantics

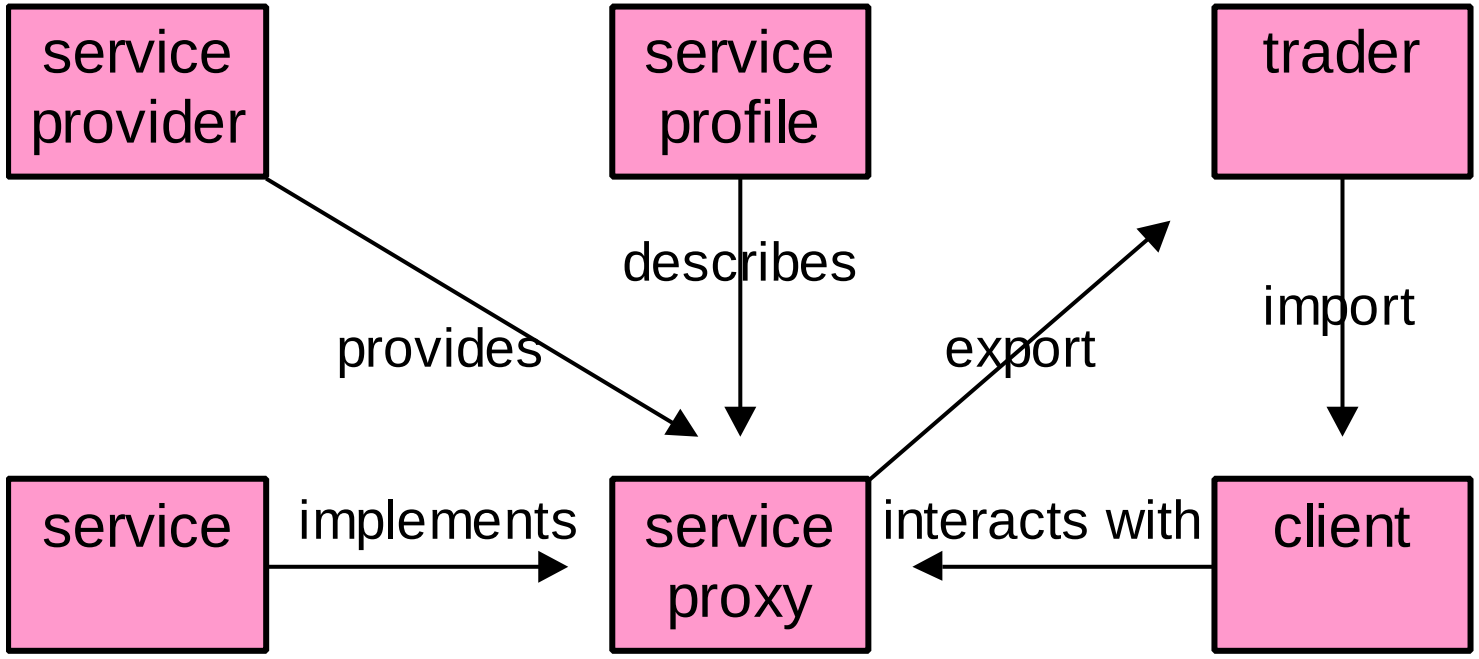
Basic Service Interaction

- Service interaction can be arbitrarily complex
  - ⇨Service provider provides suitable clients
  - ⇨Missions defined by service providers
- 3rd party missions
  - ⇨ Missions encompassing more than one service type

Services, Trading & Proxies

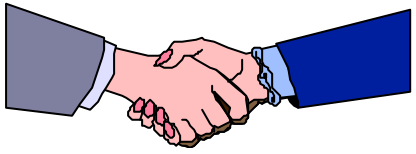
- Traders required as an infrastructure component
  - ⇨Hold service interface references
  - ⇨Hold agent missions
    - ⇨*Essentially the source of mission scripts*
- Envisage all services having proxies which inherit from Mobile Object class
  - ⇨*Capable of movement too*

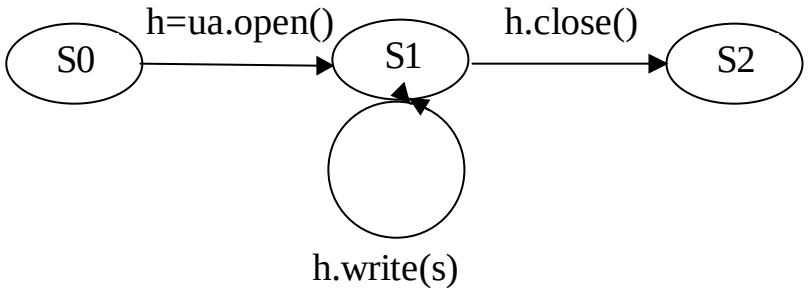
Domain Object Model



Service Contracts

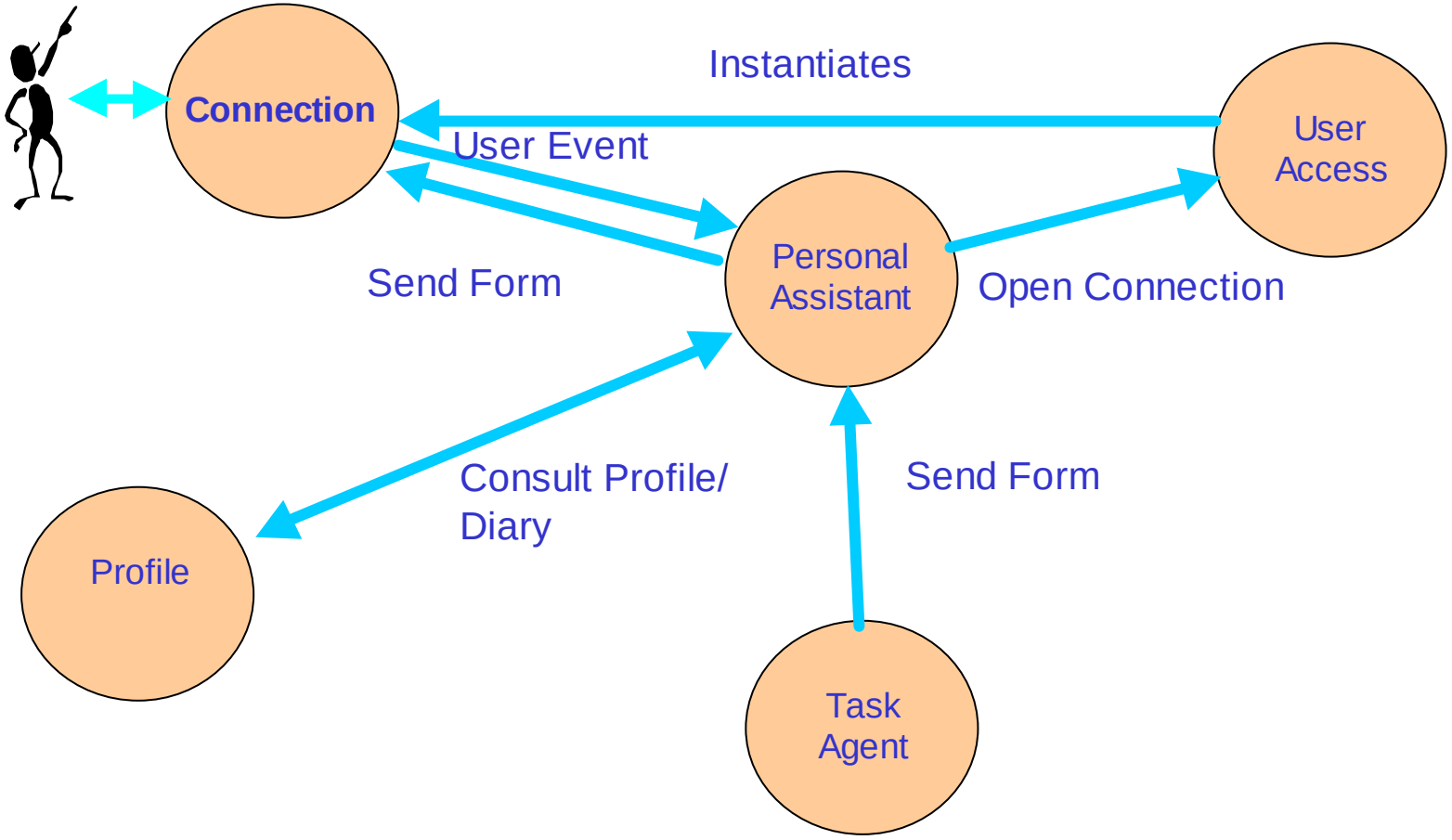
- Research into specification of agent behaviour and hence goal directed agents
  - ⇨Example:





⇨`(S0,h=ua.open(),S1),`
`(S1,h.write(s),S1), (S1,h.close,S2).`

Agent Framework - summary



Live Example...

- Using Personal Profiles 
  Package

  ⇨Mission Script 
  Package

Live Example...

- Tamagotchi 
Package

⇨Mission Script 
Package

Summary

- Mobile agents and mobile users
- Agent Framework consists of mission scripting, personal profiles and service interaction
- Implemented on Flexinet/MOW/IS from ANSA
- Pilot applications from FAST and TCM
- Further application scenarios