Preparation

1) Ensure you have a valid installation of Java (JRE or JDK of 1.5.x with JAVA_HOME set in the environment and %JAVA_HOME%\bin included in PATH).

2) Ensure you have a valid installation of Apache Ant (1.6.5 has been tested, again its 'bin' folder must be in the PATH variable).

3) Unzip HICSS06-handson.zip to a convenient location.

4) Copy the code folder from 'IRS-III\Lisp' to 'c:\users\jbd2' (i.e. forming 'c:\users\jbd2\code' etc.).

5) Launch all tools by executing 'tutorial.bat', when the IRS Visualiser is shown, navigate to IRS Browser, accept the default server settings (localhost:3000), and log-in as wsmo:wsmo.

6) Finally,navigate to the Web Services Modelling Toolkit (WSMT).

Exercise 1

The point of this exercise is to show the logical separation of goals from services, and their connection through WG-mediators.

The first stage in the virtual travel agency example concerns the provision of timetable information for european rail travel. For the purposes of the tutorial the service will be implemented by a LISP function, 'get-train-times' (p155).

1) In WSMT, right click on the 'exercise1' project, in the navigator on the left, and select 'New->Other->WSMO->Goal', enter filename 'get-train-times-goal', identifier 'get-train-times-goal', and select the 'wsml-core' variant.

(Tip: Since filenames will usually match identifiers, it is useful to copy this into the clipboard and paste for the identifier.)

2) Double click on 'get-train-times-goal.wsml' and the WSMO Visualiser will show the goal in the centre.

3) Right-click on the goal and select 'Add->Capability'; call the new capability 'get-train-times-capability'.

3) Right-click on the capability and select 'Add->Preconidition'; call the new precondition 'get-train-times-inputs'.

4) Right-click on the precondition and select 'Add->Logical Expression'; enter the following:

```
  ?origin memberOf _"http://irs.open.ac.uk/tutorial#city"
  and
  ?destination memberOf _"http://irs.open.ac.uk/tutorial#city"
  and
  ?date memberOf _"http://irs.open.ac.uk/tutorial#date"
```

5) Right-click on the capability and select 'Add->Postcondition'; call the new postcondition 'get-train-times-outputs'.

6) Right-click on the postcondition and select 'Add->Logical Expression'; enter the following:

```
  ?timetable memberOf _"http://irs.open.ac.uk/tutorial#string"
```

7) Save the goal (File->Save)

(Tip: You can now right click on the goal, in the navigator, and select 'Open With->WSMO Text Editor' to see the WSML created.)

8) Right click on the 'exercise1' project, in the navigator, and select 'New->Other->WSMO->Mediator', enter filename and then identifier 'get-train-times-mediator', variant 'wsml-core', and type 'WebService-Goal Mediator' (i.e. WG-mediator).

9) Double click on 'get-train-times-mediator.wsml' and the WSMO Visualiser will show the mediator in the centre.

10) Right-click on the medaitor and select 'Add->Source Web Service' and enter 'get-train-times-goal'. (NB. In IRS WG-mediators have a goal as source and are referenced by each service that satisfies the goal.)

11) Save the mediator.

12) Right click on the 'exercise1' project, in the navigator, and select 'New->Other->WSMO->Web Service', enter filename and then identifier 'get-train-times-service' and variant 'wsml-core'.

13) Right-click on 'get-train-times-service.wsml' and select 'Open With->WSMO Editor Launcher'. This will open the service description in the WSMO Studio's form-style editor.

14) Right-click in the 'Used Mediators' box and select 'Add Mediator'; select 'get-train-times-mediator'.

15) Save the service.

16) Double-click 'import.bat' under 'exercise1' in the navigator.

17) Navigate to the IRS Browser and double click on the ontology 'european-travel-service-descriptions', where the three new items should appear.

(Tip: If the ontology was already open at the time of import, close and reopen it by double-clicking)

18) Select 'get-train-times-service' and then (from the menu) 'Goal/WS/Mediator->Edit Goal/WS/Mediator Description...'.

19) Select the Interface tab, and then in the 'Grounding' field (on the 'Choreography' subtab) enter the following (and save):

```
  ((normal get-train-times))
```

20) From the menu, select 'Publish->Lisp Function' and publish with the default settings.

21) Finally, to test the above, select 'get-train-times-goal' and, from the menu, select 'Invoke->Achieve Goal...', entering the following:

```
  origin: milton-keynes
  destination: london
  date: (02 01 2005)
```

Exercise 2

The point of this exercise is to show how the logical separation of goals from services allows capability-driven execution.

This part of virtual travel agency example concerns the provision of booking services for european rail travel. For the purposes of the tutorial the services will be implemented by LISP functions 'book-english-train-journey' and 'book-french-train-journey' (p155).

1) In WSMT, open (click the '+' sign) and then right click the 'exercise2' project, in the navigator on the left, and select 'New->Other->WSMO->Goal', enter filename and then identifier 'book-train-goal', and select the 'wsml-core' variant.

2) As before, open the goal in the visualiser and add a capability 'book-train-capability'.

3) Add a precondition 'book-train-inputs', and to this a logical expression as follows:

```
  ?passenger memberOf _"http://irs.open.ac.uk/tutorial#person"
  and
  ?origin memberOf _"http://irs.open.ac.uk/tutorial#city"
  and
  ?destination memberOf _"http://irs.open.ac.uk/tutorial#city"
  and
  ?date memberOf _"http://irs.open.ac.uk/tutorial#list-date-and-time"
```

4) Add a postcondition 'book-train-outputs', and to this a logical expression as follows (and then save the goal):

```
  ?confirmation memberOf _"http://irs.open.ac.uk/tutorial#string"
```

5) Add a WG-mediator called 'book-train-mediator' and add 'book-train-goal' as its source (remembering to save).

6) Add services named 'book-english-train-service' and 'book-french-train-service', each with 'book-train-mediator' in 'Used Mediators' (using the Editor Launcher, and then save each).

7) Double click 'import.bat' and switch to the IRS Browser (reopening the ontology as explained above).

8) Add the grounding to 'book-english-train-service' as above and then, on the main 'Capability' tab, enter the following assumption:

```
  (kappa (?goal)
    (and
      (is-in-country (wsmo-role-value ?goal 'origin) 'england)
      (is-in-country (wsmo-role-value ?goal 'destination) 'england)))
```

9) Publish 'book-english-train-service'.

10) Add the grounding to 'book-french-train-service', add the following assumption and then publish:

```
  (kappa (?goal)
    (and
      (is-in-country (wsmo-role-value ?goal 'origin) 'france)
      (is-in-country (wsmo-role-value ?goal 'destination) 'france)))
```

11) Test the goal with the following inputs:

```
  passenger: john
  origin: milton-keynes
  destination: london
  date: (00 00 09 02 01 2005)
```

(Tip: Switching to the visualiser allows you to see which service is being invoked.)

12) Test the goal with the following inputs:

```
  passenger: john
  origin: paris
  destination: lyon
  date: (00 30 19 02 01 2005)
```

Exercise 3

The point of this exercise is to show how data mediation can play a part in the mediation between a goal and a service with disimilar input.

This part of virtual travel agency example extends the provision of booking services for european rail travel. For the purposes of the tutorial the services will be implemented by LISP functions 'book-german-train-journey' and 'mediate-time' (p156-7).

1) In WSMT, open the 'exercise3' project, then copy across 'book-train-goal.wsml' from 'exercise2' (right click on the original and select 'Copy', then right click on 'exercise3' and select 'Paste').

2) Create a new WG-mediator 'book-train-mediated-mediator' with a source of 'book-train-goal' and a mediation service 'mediate-time-goal' (and save).

3) Create a new service 'book-german-train-journey', add this mediator above and save.

4) Create a new goal 'mediate-time-goal' and add a capability 'mediate-time-capability'.

5) Add a precondition 'mediate-time-inputs' and add to it a logical expression as follows:

```
  ?date memberOf _"http://irs.open.ac.uk/tutorial#list-date-and-time"
```

6) Add a postcondition 'mediate-time-outputs' with the logical expression as follows, and save:

```
  ?mediated memberOf
  _"http://irs.open.ac.uk/tutorial#universal-date-and-time"
```

7) Add a WG-mediator 'mediate-time-mediator' with this goal as source and save.

8) Add a service 'mediate-time-service', add this mediator to 'Used Mediators' and save.

7) Double click 'import.bat' and switch to the IRS Browser (reopening the ontology there).

8) Edit 'mediate-time-goal' and, on the 'Input and Output' tab, change the 'SOAP Type' of the output from 'xml' to 'float'.

8) Add the grounding to 'mediate-time-service' and publish.

9) Test the 'mediate-time-goal' with the following input:

```
  (00 00 09 02 01 2005)
```

10) Edit 'book-train-mediated-mediator' and change 'Parent' from 'mediator' to 'wg-mediator'.

11) Add the grounding to 'book-german-train-service', add the following assumption and then publish:

```
  (kappa (?goal)
    (and
      (is-in-country (wsmo-role-value ?goal 'origin) 'germany)
      (is-in-country (wsmo-role-value ?goal 'destination) 'germany)))
```

12) Test the goal with the following inputs:

```
  passenger: john
  origin: munich
  destination: berlin
  date: (00 30 19 02 01 2005)
```