AAAI 2006

The Twenty-First National Conference on Artificial Intelligence

SWS Systems Tutorial: Hands-On Session

Barry Norton
Knowledge Media Institute,
Open University, UK

July 2006

Stage 0 - Preparation

Installation

1) Unzip AAAI06HandsOn.zip, or copy the contents of the CD, to a convenient location

Invocation

2) Double-click on 'tutorial.bat', launching the IRS-III Server, Publisher and Visualizer, and WSMO Studio 0.4.1

Creating the Project

3) Locate WSMO Studio and select (from the menu) 'File->New'

4) Choose 'WSMO Project' from the group 'WSMO', and click 'Next'

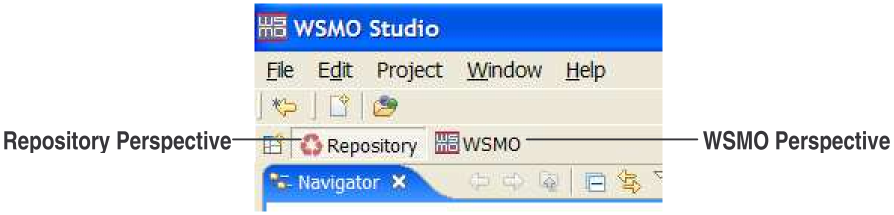
5) Enter a project name 'AAAI06', and click 'Next' (not 'Finish')

6) Enter a default namespace 'http://irs.open.ac.uk/europeanTravelServiceDescriptions#', and click 'Finish'

7) Allow the project to be associated with the WSMO Perspective by clicking 'Yes'

Stage 1 - Retrieving the Domain Ontologies

1) Go to Repository Perspective by clicking the Repository button in the toolbar (see diagram):



(If no Repository button is shown, click on the leftmost button to add a perspective, click 'Other...' and then select 'Repository')

Note that WSMO Perspective was selected in the previous stage, and that the WSMO Navigator is now replaced by the Repositories Explorer

2) Right-click in the Repositories Explorer (at the bottom left-hand corner) and select New Repository

3) Enter Name 'IRS' and select 'IRS-III Repository Adapter'

4) Double-click the new icon in the Repositories Explorer and enter the following details:

Host: localhost
Port: 3000
Service Path: soap
Username: wsmo
Password: wsmo

The Repository view will now show all ontologies, goals, mediators, and web services published in the IRS whose details were entered.

5) Locate and right-click on 'http://irs.open.ac.uk/europeanTravelServiceDescriptions#' (the ontology we're extending, entered above as default namespace), and select 'Save in Workspace'

6) Select the project created above as the parent folder, enter a file name 'europeanTravelServiceDescriptions.wsml', and click 'Finish'

7) Double-click on 'europeanTravelServiceDescriptions.wsml' in the 'Navigator' and observe, on the ‘Imports Ontologies’ box in the Ontology Editor, that this imports the ontology ‘europeanTrainTravelApplication’

8) Switch back to the Repository View (the tab ‘Repository – IRS’), locate 'http://irs.open.ac.uk/europeanTrainTravelApplication#' and save this as above (using the name europeanTrainTravelApplication.wsml) and open this

9) Switch back to the WSMO Perspective (see diagram on the left), and observe in the WSMO Navigator (in the bottom left-hand corner) that this ontology defines the concept ‘person’ (and others)

10) Using the Repository View, as above, save into the workspace the ontologies imported by europeanTrainTravelApplication, i.e.: germanCitiesKb, frenchCitiesKb and ukCountyKnowledgeBase

11) In the final case, which takes some time, right click on the ontology in the WSMO Navigator and select ‘Ontology Contents Info’ – observe that more than 1600 instances have been transferred

12) Finally, observe that ukCountiesKnowledgeBase imports ‘commonConcepts’, so save this in the workspace. Observe that ‘location’ has a subconcept ‘geographicalRegion’, and that this subclasses into ‘geopoliticalEntity’ then ‘city’

Stage 2 – A Simple Timetable Service

Creating the Communication Model

1) Move back to the Ontology Editor for europeanTravelServiceDescriptions, right click on the name in the WSMO Navigator, and select ‘Add Concept’

2) Enter a concept ID of 'reqGetTimetable' and click 'OK'

3) Double-click 'reqGetTimetable' in the WSMO Navigator and then right-click in the Attributes pane of the new Concept Editor and select 'Add Attribute'

4) Enter an attribute name 'origin' and click 'OK'

5) Double-click on 'origin [0..*]', where it appears under Attributes, and select 'set MIN', next to 'Cardinality:' in 'Attribute Properties' and enter '1' in the edit box, then select 'set MAX' and enter '1' there

6) Right-click in 'Range Concepts' and select 'Add Concept'

7) Select 'commonConcepts -> location -> geographicalRegion -> geopoliticalEntity -> city', and then click 'OK'

8) Repeat this process (steps 8-12) to add an attribute 'destination' of the same cardinality and concept

9) Add a third attribute named 'dateOfTravel', set the cardinality as exactly 1 as before, but this time set the range to the data type (i.e. select 'Add Data Type' from the context menu for Range Concepts) 'http://www.wsmo.org/wsml/wsml-syntax#date'

10) Add another concept to the europeanTravelServiceDescriptions ontology called 'resGetTimetable', open this and add an attribute (as in steps 6-8), call this attribute 'timetable', of cardinality 1, attach (data) type 'http://www.wsmo.org/wsml/wsml-syntax#string' (cf. step 14)

11) Save the ontology by clicking 'File -> Save' from the menu

Creating the Goal

12) Right-click on ‘ESWC06’ in the Navigator (at the top right-hand corner of the screen), select ‘New -> Other...’ and then ‘WSMO -> Goal’

13) Enter a filename 'goalGetTimetable.wsml' and then click 'Next'

14) Enter a goal identifier 'goalGetTimetable', accept the default namespace, and select the 'wsml-core' WSML variant, then click 'Finish'

15) Double-click on 'goalGetTimetable.wsml' in the Navigator

16) In the new Goal Editor, right-click in the 'Import Ontologies' pane, then select 'Add Ontology'

17) Select 'europeanTravelServiceDescriptions' by double-clicking

18) Right-click in the 'Interfaces' list, under 'Capability & Interfaces', and select 'Create Interface'

19) Enter an identifier 'intGoalGetTimetable' and click 'OK'

20) In the WSMO Navigator, open the definition of goalGetTimetable (by clicking the '+' symbol), then right-click on intGoalGetTimetable and select 'Create Choreography'

21) Enter an identifier ‘chorGoalGetTimetable’ and click ‘OK’

22) Right-click on ‘chorGoalGetTimetable’ and select ‘Create State Signature’

23) Enter an identifier ‘ssGoalGetTimetable’ and click ‘OK’

24) Double-click on ‘ssGoalGetTimetable’, then right-click within the 'IN modes' tab and select 'Add Mode'

25) Select 'europeanTravelServiceDescriptions -> resGetTimetable'

(The response is an IN Mode to the goal, because the client is receptive to it)

26) Switch to the 'OUT modes' tab, right-click and select 'Add Mode', then select 'europeanTravelServiceDescriptions -> reqGetTimetable'

(The request is an OUT mode to the goal, since it sends this)

27) Save the goal definition by clicking 'File -> Save' from the menu

Creating the Mediator

28) Create a new Mediator in a file 'wgGetTimetable.wsml', with the identifier 'wgGetTimetable', in the default namespace and the wsml-core variant, as with the goal in steps 12-14, but select the Mediator Type 'WebService-Goal Mediator' (wg-mediator) before clicking 'Finish'

29) Double-click on 'wgGetTimetable' in the Navigator, right-click in 'Sources' under 'Sources & Target', in the resulting Mediator Editor, and select 'Add Source'

30) Select 'goalGetTimetable' as the source of the mediator by double-clicking

31) Save the mediator definition by clicking 'File -> Save' from the menu

Creating the Service

32) Create a new Web Service, in a file 'wsGetTimetable.wsml', called 'wsGetTimetable', in the default namespace and the wsml-core variant, as above

33) Double-click on 'wsGetTimetable' in the Navigator, and import the ontology 'europeanTravelServiceDescriptions'

34) Right-click in 'Used Mediators', select 'Add Mediator', then select 'wgGetTimetable' by double-clicking

35) Add a new interface 'intWsGetTimetable', and then a new choreography 'chorWsGetTimetable' with state signature 'ssWsGetTimetable', as in steps 18-23

36) Add the modes for chorWsGetTimetable, as for chorGoalGetTimetable in steps 24-26, but this time with 'reqGetTimetable' as the IN mode, and 'resGetTimetable' as the OUT mode

37) Right-click on 'reqGetTimetable' on the IN modes tab and select 'Add Grounding'

38) Enter the Grounding IRI 'irslisp://localhost:3001/soap/europeanTravelServiceDescriptions/getTrainTimes' and click 'OK'

39) Save the service definition by clicking 'File -> Save' from the menu

40) Switch back to the 'IRS' Repository View (the tab opened in step 11)

41) Drag 'goalGetTimetable.wsml' then 'wgGetTimetable' and then 'wsGetTimetable' onto this view (it is not necessary to choose the appropriate pane)

Invoking the Goal

42) Return to the Ontology Editor for europeanTravelServiceDescriptions (this can be achieved by double-clicking its file in the Navigator)

43) Right-click on ‘reqGetTimetable’ and select ‘Add Instance’

44) Enter an identifier ‘testGetTimetable’ and select ‘OK’

45) Double-click on ‘testGetTimetable’ to open its Instance Editor

46) Right-click in ‘Attribute Values’, select ‘New Attribute Instance Value’, and then select ‘europeanTravelServiceDescriptions -> reqGetTimetable -> origin’

47) In the ‘Select Attribute Instance Value’ dialog, select ‘ukCountyKnowledgeBase -> englishTown -> york’

48) Repeat steps 46 and 47 to add the instance ‘ukCountyKnowledgeBase -> englishTown -> london’ to attribute ‘destination’

49) Right-click in ‘Attribute Values’, select ‘New Attribute Data Value’ (remembering dateOfTravel is of a data type), and select dateOfTravel

50) Enter a value of ‘2006, 7, 18’ (note: do not reproduce the quotes)

51) Save this new definition, by selecting 'File -> Save' from the menu

52) Switch back to the Repository Perspective (cf diagram, Stage 1)

53) Right-click on 'IRS' in the Repositories Explorer and select 'Achieve Goal'

54) Click 'Select' beside 'Goal' and select 'goalGetTimetable' by double-clicking

55) Click 'Add' in 'Input Instances' and select 'europeanTravelServiceDescriptions -> reqGetTimetable -> testGetTimetable' by double-clicking

56) Click execute and an appropriate timetable should be shown

54) Cancel the Achieve Goal dialog, close all editors (as in step 46), and switch back to the WSMO Perspective

Stage 3 – A Goal met by multiple Services

Creating the Communication Model

1) Navigate back to europeanTravelServiceDescriptions, and the WSMO Perspective, and create a new concept 'reqBookTrain', with attributes:

```
  'passenger', of concept 'europeanTrainTravelApplication -> person';
  'origin' and 'destination', of concept 'city' (as before);
  'timeOfTravel' of data type '...wsml-syntax#dateTime' (note: not date)
```

2) Add concept 'resBookTrain', with attribute 'confirmation' of WSML type 'string'. (Remember to save both definitions.)

3) Add three instances to reqBookTrain, as follow:

```
  instance: testBookEnglishTrain
    passenger: ‘europeanTrainTravelApplication->person->academic->barry’
    origin: ‘ukCountyKnowledgeBase->englishTown->york’
    destination: ‘ukCountyKnowledgeBase->englishTown->london’
    timeOfTravel: 2006,7,18,17,0,0

  instance: testBookFrenchTrain
    passenger: ‘europeanTrainTravelApplication->person->student->john’
    origin: ‘frenchCitiesKb->frenchCity->toulouse’
    destination: ‘frenchCitiesKb->frenchCity->paris’
    timeOfTravel: 2006,7,18,14,30,0

  instance: testBookGermanTrain
    passenger: ‘...->person->businessPerson->liliana’
    origin: ‘germanCitiesKb->germanCity->walldorf’
    destination: ‘germanCitiesKb->germanCity->frankfurt’
    timeOfTravel: 2006,7,18,15,0,0
```

(Remember to save these definitions)

Creating the Goal

4) Create a new goal 'goalBookTrain', importing this ontology, then add an interface 'intGoalBookTrain', and a choreography 'chorGoalBookTrain' with a state signature 'ssGoalBookTrain', importing the communication concepts created above.

(Remember that the response is the IN mode, as in Stage 2, step 26)

Creating the Mediator

32) Create a new Mediator 'wgBookTrain', with this new goal as source.

Creating the Services

33) Create a new Web Service, 'wsBookEnglishTrain', double-click and import the ontology ‘europeanTravelServiceDescription’ and using mediator ‘wgBookTrain’, then add an interface 'intWsBookEnglishTrain', a choreography 'chorWsBookEnglishTrain', and a state signature 'ssWsBookEnglishTrain' with the modes reversing those of the goal

34) Ground the request (as in Stage 2, Step 38) to 'irslisp://localhost:3001/soap/europeanTravelServiceDescriptions/bookEnglishTrainJourney'

35) Create a capability for this service by right-clicking 'wsBookEnglishTrain' in the WSMO Navigator and selecting 'Create Capability', call this 'capBookEnglishTrain'

36) Double-click to open the Capability Editor, then right-click in Assumptions and select 'Create Axiom'

37) Enter a name 'http://irs.open.ac.uk/europeanTravelServiceDescriptions#aspBookEnglishTrain', and double-click 'aspBookEnglishTrain' to open the Axiom Editor.

38) Right-click in 'Logical Expressions' and select 'Add Expression', then click in the Expression Editor and enter the following:

```
        ?asp[hasOCMLLogicalExpression hasValue "(kappa (?goal)
          (and
            (is-in-country (wsmo-role-value ?goal 'origin) 'england)
            (is-in-country (wsmo-role-value ?goal 'destination)
              'england)))"]
          memberOf OCMLAssumption.
```

39) Save this service, and create a similar one for a French service, changing the naming accordingly (and changing 'england' to 'france' in the assumption), grounded to 'irslisp://localhost:3001/soap/europeanTravelServiceDescriptions/bookFrenchTrainJourney', remembering to save this too.

40) Switch to the Repository Perspective and open the IRS Repository View (by double-clicking) and export to IRS (note: in this order exactly): 'goalBookTrain', 'wgBookTrain', 'wsBookEnglishTrain', then 'wsBookFrenchTrain', as before

41) Invoke the goal, as before, trying the result of each of the three request instances produced above

Stage 4 – A Service with Data Mediation

Extending the Communication Model

1) Re-open 'europeanTravelServiceDescriptions' and add a new concept 'reqBookGermanTrain', matching the other request except that 'timeOfTravel' is given the WSML type 'integer'

(A new German service will represent the date and time in a single integer, and it is necessary to create a mediation goal for express the need to mediate between these two representations, and a service that can achieve this.)

2) Add another concept 'reqMediateTime', with an attribute 'timeOfTravel' (name must match 'goalBookTrain') of WSML type 'dateTime', and 'resMediateTime', with an attribute 'mediatedTimeOfTravel' of WSML type 'integer'

Creating the Mediation Goal and Service

3) Create, and save, a new goal 'goalMediateTime', importing 'europeanTravelServiceDescriptions', with a choreography where the state signature attaches this request and response, as usual

4) Create, and save, a new wg-mediator 'wgMediateTime' with this goal as source

5) Create, and save, a new web service description 'wsMediateTime', which uses this mediator, reverses the goal’s modes, and has the IN mode grounded to 'irslisp://localhost:3001/soap/europeanTravelServiceDescriptions/mediateTime'

Creating the Mediator and Service

6) Create a new wg-mediator 'wgBookGermanTrain', with 'goalBookTrain' as source, but also include an entry in 'Service' of 'http://irs.open.ac.uk/europeanTravelServiceDescriptions#goalMediateTime', then save

7) Create a web service description 'wsBookGermanTrain', which follows the definition of the English and French services, but uses the mediator 'wgBookGermanTrain' and the request 'reqBookGermanTrain', remembering to save, then export through the Repository View, in the order: 'goalMediateTime', 'wgMediateTime', 'wsMediateTime', 'wgBookGermanTrain', 'wsBookGermanTrain'

Testing the Whole Scenario

8) As before, 'goalBookTrain' with each of the three requests in turn.

We should like to acknowledge the integration work of Barry Norton and Carlos Pedrinaci, from KMi, and Marin Dimitrov and Alex Simov, from OntoText. This tutorial was developed from an earlier tutorial, based wholly in the IRS, by Barry Norton.

For further information on IRS-III, please visit
http://kmi.open.ac.uk/projects/irs

For further information on WSMO Studio, please visit
http://www.wsmostudio.org