Assignment 6

Using GridNexus to Create Workflows that use Web and Grid Services

Authors: Dr. Clayton Ferner and Dr. Barry Wilkinson

March 26, 2010

In this assignment, you will create several workflows that will use already written and deploy web services on the servers. As you prepare your Word document showing that you accomplished the tasks, you should include pictures of the workflow, pictures of the results (the JXPL Display), and a copy of the JXPL script. The JXPL script can be generated by right-clicking on the JXPL Display and turning off the flag "Evaluate JXPL" and "Output in Text". When you run the workflow, it will not do anything other than produce the JXPL script that would carry out the workflow. Cut and paste this into your document. Don't forget to turn these flags back on afterwards so it will actually do the work.

Task 1: Setup

a. If you are using the computer on which you did Assignment 3, then GridNexus should be setup and ready to run. If not, then you will need to perform the setup task 1 from Assignment 3.
b. Create a new proxy on the local machine (Start->All Programs->GridNexus->Utils->Proxy->grid-proxy-init).

Task 2: Setting up GridNexus for Web Services (10 points)

a. Log into torvalds.cis.uncw.edu using ssh and locate the jar files in the directory in $GLOBUS_LOCATION/lib. The jar files that you will need to work with the web services in this assignment are:

   org_globus_examples_services_core_first.jar
   org_globus_examples_services_core_first_stubs.jar
   org_globus_examples_services_core_second.jar
   org_globus_examples_services_core_second_stubs.jar
   org_globus_examples_services_core_third.jar
   org_globus_examples_services_core_third_stubs.jar
   org_globus_examples_services_core_C3M5P1.jar

org_globus_examples_services_core_C3M5P1_stubs.jar
org_globus_examples_services_core_C6M2P5V1.jar
org_globus_examples_services_core_C6M2P5V1_stubs.jar

b. Determine the name of the Addressing Locator class of the service by executing the following command:

```
jar -tf org_globus_examples_services_core_first_stubs.jar
```

The output should look something like:

```
META-INF/
META-INF/MANIFEST.MF
org/
org/globus/
org/globus/examples/
org/globus/examples/stubs/
org/globus/examples/stubs/MathService_instance/
org/globus/examples/stubs/MathService_instance/bindings/
org/globus/examples/stubs/MathService_instance/service/
org/globus/examples/stubs/MathService_instance/AddResponse.class
org/globus/examples/stubs/MathService_instance/GetValueRP.class
org/globus/examples/stubs/MathService_instance/MathPortType.class
org/globus/examples/stubs/MathService_instance/MathResourceProperties.class
org/globus/examples/stubs/MathService_instance/SubtractResponse.class
org/globus/examples/stubs/MathService_instance/bindings/MathPortTypeSOAPBindingStub.class
org/globus/examples/stubs/MathService_instance/service/MathService.class
org/globus/examples/stubs/MathService_instance/service/MathServiceAddressing.class
org/globus/examples/stubs/MathService_instance/service/MathServiceAddressingLocator.class
org/globus/examples/stubs/MathService_instance/service/MathServiceLocator.class
```

The Addressing Locator class is the class name that contains (obviously) the words "AddressingLocator" with the slashes (/) replaced with periods (.). So for the above output, the class name is :

org.globus.examples.stubs.MathService_instance.service.MathServiceAddressingLocator

This class name will be needed to configure the WSRF Client in GridNexus, so you should make a note of it. ***Do this for all of service stub files above and write these class names down as you will need them later.*** *Also include this list in your submission document.*

c. Download the jar files (using sftp or WinSCP or some such file transfer) to your machine. Place them in C:\Program Files\GridNexus2.02\lib.

Task 3: Create a Grid Service Workflow (40 points)

a. Start GridNexus. (It is important that you restart GridNexus if you should add any new jars to the lib because it needs to re-read the lib directory.)

b. Create a new workflow. Drag a WSRF Client module (Module Library->Transformers->Grid->WSRF Client) onto the workflow.

c. Right click on the WSRF Client and configure it by providing:
   - the URL of the grid service. This should be "https://torvalds.cis.uncw.edu/wsrf/services/examples/core/first/MathService". Be sure to include the quotations.
   - the Addressing Locator class name for the first web service from Step 2b (e.g. "org.globus.examples.stubs.MathService_instance.service.MathServiceAddressingLocator"). Again, be sure to include the quotations.

   If you have given the Addressing Locator class correctly and the correct jar has been placed in C:\Program Files\GridNexus2.01\lib, then the WSRF Client will should show the operation names ("getValueRP", "add", and "subtract" as well as others) on the input ports. If it does not configure itself, then either the jars are not in the GridNexus lib directory, you didn't restart GridNexus, or the Addressing Locator class name is incorrect. **IT IS VERY IMPORTANT NOT TO MAKE ANY TYPING MISTAKES HERE.**

d. Right click on the WSRF Client and choose Customize Name. Rename the module "First Math Service Client"

e. Right click on the Client module and choose "Save Actor in Library". This will bring up the user library in a new window with the new module copied to it. Save the Library and close it. You should now see the new actor appear in the "user library" in the list of libraries on the left panel of GridNexus like the figure below:

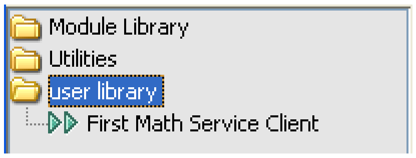

Figure 1: Math Client saved in the user library

f. Go back to your workflow and drag two Const module and another "First Math Service Client" module from the user library to the workflow. Set the value of one Const module to be some integer value (without quotes) and then connect this to the add operation of one Math Client.

g. Set the second Const module to the value **"void"** (including the double quotes). Connect this Const module to the getValueRP operation of the other Math Client.

h. Drag a Prog module (Module Library->Transformers->Basic->primitives->Prog) and JXPL Display module (Module Libary->Sinks->JXPL Display) to the workflow. Connect the first Math Client (the one with the constant integer connected to its add port) to the Prog, then connect the other Math Client to the Prog.

i. Connect the Prog to the JXPL Display. Save the workflow. It should look something like Figure 2.
j. Save the workflow then run the workflow. If it runs correctly, the result should be an integer value. **Is the result what you expected? If not, then why would it be something other than what you expect? Run it several times to see what happens.**
k. In your submission document, include:
    1. your answers or comments to the above questions;
    2. screen shots of the workflow and the JXPL display; and
    3. the JXPL of your workflow. (Right-click on the JXPL display, choose configure, turn off "Evaluate JXPL", and "Output in Text"; then re-run the workflow. Copy the JXPL into your submission document. Make sure you include ALL of the JXPL.)

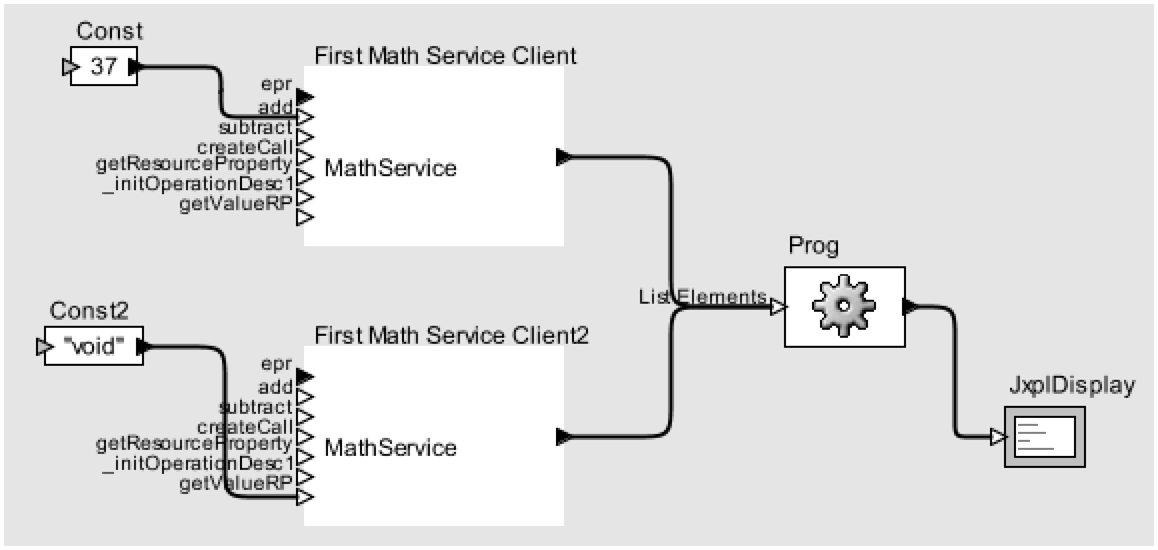

Figure 2: Math Client first workflow

Task 4: Back to the Flower Bed (30 points)

Step 1:

a. Create a new workflow. Drag a WSRF Client module onto the workflow.
b. Right click on the WSRF Client and configure it by providing:
    - the URL of the trapezoidal integration service. This should be "https://coit-grid02.uncc.edu:8440/wsrf/services/examples/core/second/ MathService". (Don't forget the port number.)
    - the Addressing Locator class name from Step 2b for the second Service.

c. Rename this module "Second Math Service Client"
d. Right click on the WSRF Client module and choose "Save Actor in Library". This will bring up the user library in a new window with the new module copied to it. Save the Library and close it. You should now see the new actor appear in the "user library" in the list of libraries on the left panel of GridNexus.
e. Add another "Second Math Service Client" module from the library to the workflow, as well as three Const modules, a Prog, and a JXPL Display.
f. Give two of the Const boxes floating point values (without quotes) and the third Const box the value of **"void"** (with quotes). Connect the Const boxes with the numbers to the setA and setB inputs of the first client.
g. Connect the the "**void**" Const box to the getaRP input of the other client.
h. Connect the two clients to the Prog. (First the one with the numbers given to setA and setB, then the one with "void" being given to getaRP.) Connect the Prog to the JXPL Display. The workflow should look something like figure 3.
i. Save the workflow and run it. If it works correctly, then you will get value you set for A. Reconnect to "void" constant to the getbRP input port and rerun. This time you should get the value you assigned to B.
j. Save the workflow and run it. **Did it produce the expected result?**
k. Change the values of the two Const boxes to be the A and B values used to compute the area of the flower bed from assignment 1.
l. Add another Const box to the workflow and give it a large integer value. Connect this new Const box to the "integrate" input port of the first client. This large integer is the number of trapezoids to use in estimated the area of the flower bed.
m. Connect the "void" Const box to the "getAreaRP" input port of the second client.
n. Save the workflow and run it. **Did it produce the expected result?**
o. In your submission document, include:
   1. your answers or comments to the above questions;
   2. screen shots of the workflow and the JXPL display;
   3. the JXPL of your workflow.

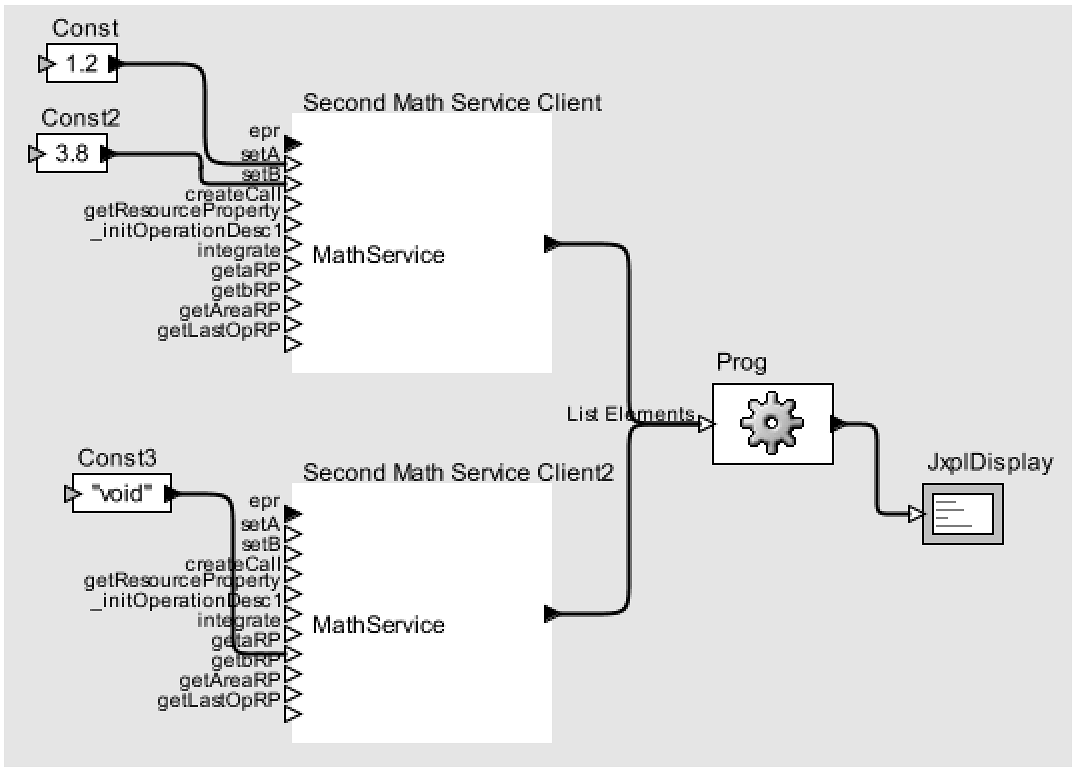

Figure 3: Workflow that sets up the Integration

Step 2:

a) Create a new workflow. Drag a WSRF Client module onto the workflow.
b) Right click on the WSRF Client and configure it by providing:
    - the URL of the mulch service. This should be "https://torvalds.cis.uncw.edu/wsrf/services/examples/core/third/MathService".
    - the Addressing Locator class name from Step 2b for the third Service.
c) Rename this module "Third Math Service Client"
d) Right click on the WSRF Client module and choose "Save Actor in Library". Save the Library and close it.
e) Create a workflow similar to the one in figure 3 except that it sets the thickness and price for the mulch from assignment 1. Have the workflow return the thickness.
f) Save the workflow and run it to verify that it returns the correct value. Do it again, but return the price to make sure the correct price is set as well.
g) Save the workflow.
h) In your submission document, include:

1. your answers or comments to the above questions;
2. screen shots of the workflow and the JXPL display;
3. the JXPL of your workflow.

Step 3:

a) Create a new workflow that puts the two services together.
b) The workflow should:
   1. Set A and B of the “Second Math Service Client” to the appropriate values for the flower bed as well as set the number of trapezoids to a large integer
   2. Set the thickness and price of the mulch of the “Third Math Service Client”
   3. Call the getAreaRP of the “Second Math Service Client” and pass this result to the “mulch” input port of the “Third Math Service Client”
   4. call the “getVolumeRP” operation of the “Third Math Service Client” to get the final result.
c) The resulting workflow should look something like figure 4.
d) Save the workflow and run it. Did it produce the expected result?
i) Run it again, but this time call the getCostRP operation. Did it produce the expected result?
j) In your submission document, include:
   1. your answers or comments to the above questions;
   2. screen shots of the workflow and the JXPL display;
   3. the JXPL of your workflow.

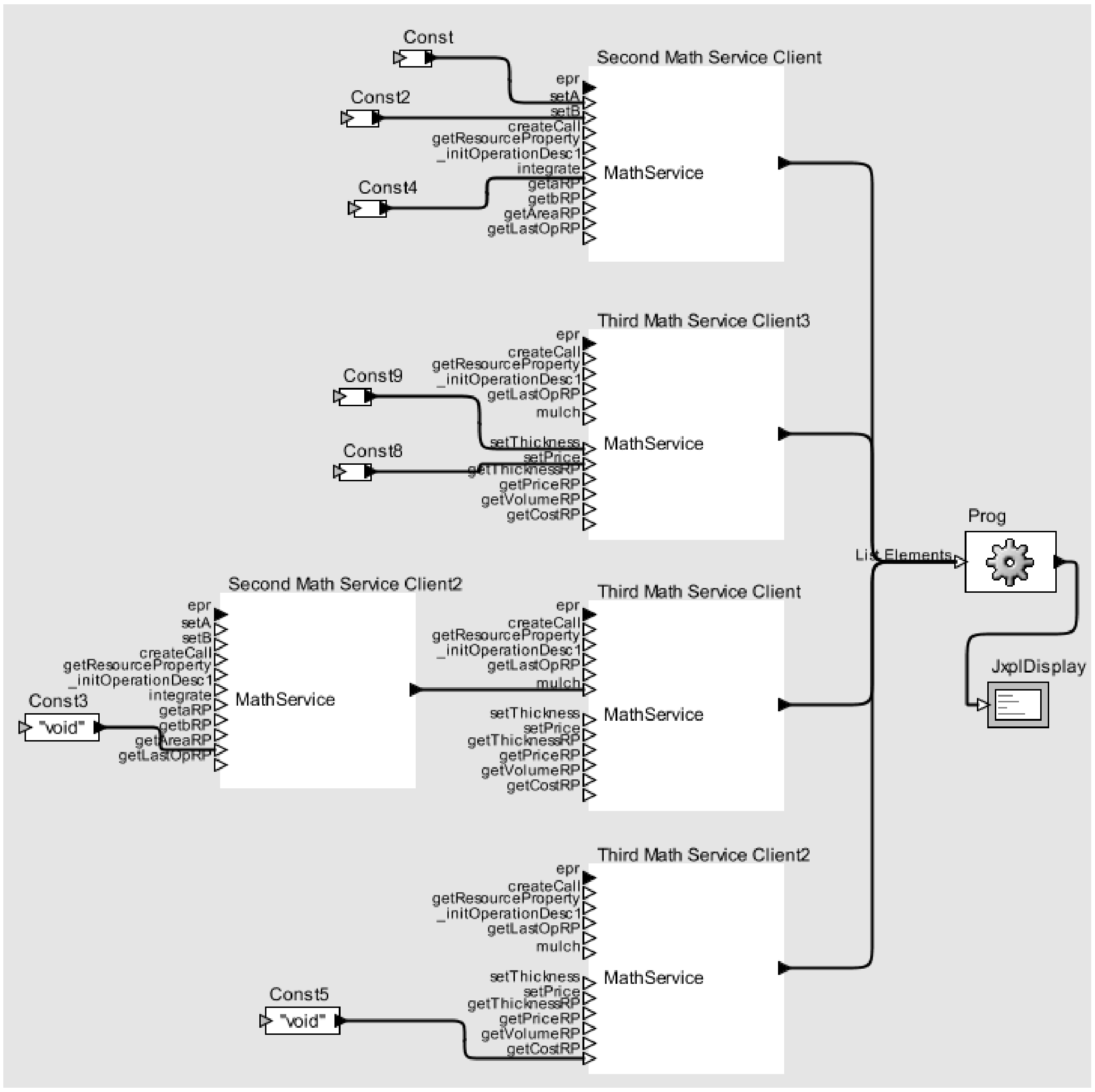

Figure 4: Workflow that puts it all together

Task 5: Disease Control (20 points)

In this tasks, you will create a workflow that does something similar to the previous task, except that the problem is different. Instead of computing the area of a flower bed and the amount of mulch needed to cover it, you will be run two models. The first model is of the interaction of a drug in the human body. The second is a model of the spread of a preventable but infectious disease. We are assuming that there is a drug available to cure the disease. The drug needs to be in the the blood stream at a minimum concentration for a minimum number of days before the person is “cured”. This minimum number of days will be used to model over time the percentage of a population that are uninfected, infected, quarantined, immune, and dead.

The workflow will be similar to the one you created in Task 4 in the sense that you will use a service on one grid resource from which output will be used as input of a service on another grid resource. You will need to configure WSRF clients to work with the services, use them to set the input values, and connect them in such as way that the information flows between them. You will take the output and create a graph of the population.

Step 1:

a. Create a new workflow. Drag a WSRF Client module onto the workflow. Configure the WSRF Client with:
   - the URL of the C3M5P1 service. This should be "https://coit-grid02.uncc.edu:8440/wsrf/services/examples/core/C3M5P1".
   - the Addressing Locator class name from Step 2b for the C3M5P1 Service.

b. Change the name of the actor and save it in the user library

c. Complete the workflow to set values for the parameters as follows:
   1. HalfLife = 22 hours
   2. Interval = 8 hours
   3. Dosage = 100,000 µg
   4. Digestion Rate 12% (.12)
   5. Absorption Fraction 12% (.12)

d. After setting these parameters, call the operation "calculation". This will produce a table of output showing time (hours) and concentration (µg/ml) of the drug in the blood streams. The drug is effective after reaching a concentration level (Minimum Effective Concentration or MEC) of 10 µg/ml, but becomes toxic after reaching a level of 20 µg/ml (Minimum Toxic Concentration or MTC)

e. Save the workflow and run it.

f. Cut the table of data from the JXPL display and paste it into something like Excel to create a scatter graph plotting the concentration against the time (hours). The concentration will fluctuate as the person takes a pill and that drug is slowly absorbed and processed by the body. This drug will "cure" the infectious disease after being in the body above the MEC for a minimum of 10 days.

g. Change the workflow to produce the recovery rate which we'll use below and save it. The recovery rate is the inverse of the number of days until the person has reached the point of being "cured".

Step 2:

a. Create a new workflow. Drag a WSRF Client module onto the workflow. Configure the WSRF Client with:
   - the URL of the C6M2P5V1 service. This should be

"https://torvalds.cis.uncw.edu/wsrf/services/examples/core/C6M2P5V1".
- the Addressing Locator class name from Step 2b for the C6M2P5V1 Service.

b. Complete the workflow to set values for the parameters as follows:
   1. b = 0.99999 (probability that a contact between person in infectious_undetected and someone in susceptible result in transmission of disease)
   2. k = 25 (mean number of contacts per day)
   3. m = 0.0975 (per capita death rate)
   4. p = 0.2 (fraction per day of exposed people who become infectious)
   5. q = 0.1 (fraction per day of individuals in susceptible who have had exposure to disease that go into quarantine)
   6. u = 0.1 (fraction per day of those in susceptible_quarantined who are allowed to leave quarantine)
   7. v = 0.4 (per capita recovery rate)
   8. w = 0.6 (rate of isolation)

c. After setting these parameters, call the operation "calculation". This will produce a table of output showing the initial population of 10500 people (10000 healthy and 500 infected but undetected).

d. Save the workflow and run it.

Step 3:

a. Create a new workflow that puts the two parts above together. The workflow should set the parameters of the C3M5P1 service as above in Step 1 and then the parameters of the C6M2P5V1 service from above in Step 2. Delete the value assigned to *v* (the recorvery rate). The workflow should then get the recovery rate from the C3M5P1 service and give that as input to the "setV" operation of the C6M2P5V1 service.

b. Finally, the workflow should call the "calculation" operation of the C6M2P5V1 service to get the output.

c. Cut the resulting output and paste it into Excel or some program that can produce a graph. Delete the "Day" in cell A1. Create a "2-D stacked area graph of the population at it moved from susceptible (healthy) into infected and then immune or dead.

d. In your submission document, include:
   1. your answers or comments to the above questions;
   2. screen shots of the workflow and the JXPL display;
   3. the JXPL of your workflow.
   4. a screen shot or copy of the graph you created of the data

6. Assignment Submission

How to submit your assignment is described in a separate document.

Submit a *single* PDF document by the due date posted on the course home page. Include:

- Snapshots of the workflow
- Snapshots of the results (JXPL Display)
- JXPL script
- Show that you successfully followed the instructions and performed all tasks.
- Conclusions
- **For the record, state what programming skills and knowledge you used but already knew before doing this assignment This information may be helpful in subsequent course developments. No information you provide will reduce your grade, but extra credit might be given for very useful comments.**