CS 410/51 Mini-Project 1 – Building a complex scientific workflow

Due Thursday, August 24

In the previous assignment you constructed a simple workflow using Kepler.
In this assignment you will construct a more complex biological workflow using Kepler and several biology web services. We will examine an existing biological workflow written in Java or Perl, and rewrite it using Kepler.

Note that this assignment is considerably harder than the last one. The purpose of the assignment is to introduce you to some of the challenges of implementing scientific workflows and get you to think about the tradeoffs between hand-coding vs. using a tool like Kepler. Please start early and ask questions if you get stuck. I will be generous with partial credit if you carefully document what each part of the workflow is supposed to do, and will give points for the parts that work correctly. I recommend using lots of Display actors to display intermediate results and help with debugging.

During the testing phase, it may help to save the web service query results to a file and replace the web service actor with a File Reader. That way you won't need to wait a few minutes for the web service to return your query results each time you test your workflow.

A written description of the workflow, along with links to source code and the Web Services used, is at http://xml.nig.ac.jp/workflow/blast_clustal.html

Download the source code in either Java or PERL (whichever language you are most familiar with. Personally I found the PERL easier to read). You are welcome to compile and run the code if you wish, but it is not required. Note that to run this code, you will need Apache axis (for Java) or SOAP:Lite (for PERL) on your machine. If you are unable to run the code on your machine, don't worry, you only need to be able to read the source code to complete this assignment.

(Note that a WSDL description of any web service <xxx> is available at http://xml.nig.ac.jp/wsdl/<xxx>.wsdl). The links on the workflow page will take you to a "Method" page that provides a query interface the method used by the given web service. Clicking on the "document" link on the page will give you documentation on other methods provided by the web service. Note that some methods have identical functionality but produce outputs in different formats, you may find these useful later.

1. (5 points) Using the query in the test.txt file (downloaded with your Java or Perl source code), use the Blast web service linked from the workflow webpage. Approximately how many distinct accession numbers (second column) are in the result? How long would it take to run GetEntry on each of these by hand?

2. (5 points) Create a blank Kepler workflow and insert a Web Service actor. Right click on the actor and select configure and type in the Blast WSDL URL (http://xml.nig.ac.jp/wsdl/Blast.wsdl) for the wsdlUrl value (leave the others blank for now). Click "commit" and wait a few seconds. Configure the actor again and look at the methodName field. Has the interface changed? How? Set the methodName to "searchParam" and click commit. What does the Web Service icon look like now?
3. 3. (80 points) Construct the BLAST-ClustalW workflow using Kepler Web Service actors for each service specified at the web site (as well as any other actors you need). You can copy the parameter settings you need for each web service from the source code (Java or PERL). Note that you can also get hints on how to parse the results returned by a web service by looking at the PERL or Java code. Many of the string and array actors that you saw in the last assignment will be useful here (especially those that use regular expressions).

   Basically your workflow should do the following:
   Given a sequence from the file test.txt:
   1. run the Blast web service on this query
   2. Extract the top K distinct accession numbers (second column) from the result and run the GetEntry web service on each.
   3. Extract the accession number, organism, and sequence from each of the K results.
   4. Construct a sequence for each in FASTA format.
   5. Paste these K sequences together to form a multiple sequence alignment query and submit to the query to the ClustalW web service.

(Note – removing duplicates in Kepler seems to be tricky. The Array Sort actor has a remove duplicates option, you are welcome to use this actor. However this actor also changes the sort order of the result returned by the BLAST web service. For the purposes of this assignment that is fine, but note that if you were using this workflow in practice you would need to write an actor or function to remove duplicates while preserving the original order)

4. (10 points) Discuss your experiences building this workflow. What parts were

hardest/easiest? Comparing your Kepler workflow to the PERL or Java source code, what are the advantages/disadvantages of each type of implementation (i.e., using a scientific workflow tool vs. coding by hand)? Your response does not need to be long, one or two paragraphs is fine.