Advanced Taverna features

Alan R Williams

materials by

Aleksandra Pawlik

Katy Wolstencroft

University of Manchester

Advanced Taverna features

- This tutorial follows on from the "*An Introduction to Designing and Executing Workflows with Taverna*" tutorial
- In this tutorial we will explore how you can alter the running of services in Taverna:
  - list handling,
  - looping,
  - control links
  - retries and
  - parallel invocation
- As in the previous tutorial, workflows in this practical use small data-sets and are designed to run in a few minutes. In the real world, you would be using larger data sets and workflows would typically run for longer

Exercise : List handling - introduction

As you have already seen, Taverna can automatically iterate over sets of data.

When 2 sets of iterated data are combined, however, Taverna needs extra information about how they should be combined. You can have:

- **A cross product** – combining every item from list 1 with every item from list 2 - ***all against all***
- **A dot product** – only combining item 1 from list 1 with item 1 from list 2, and so on – ***line against line***

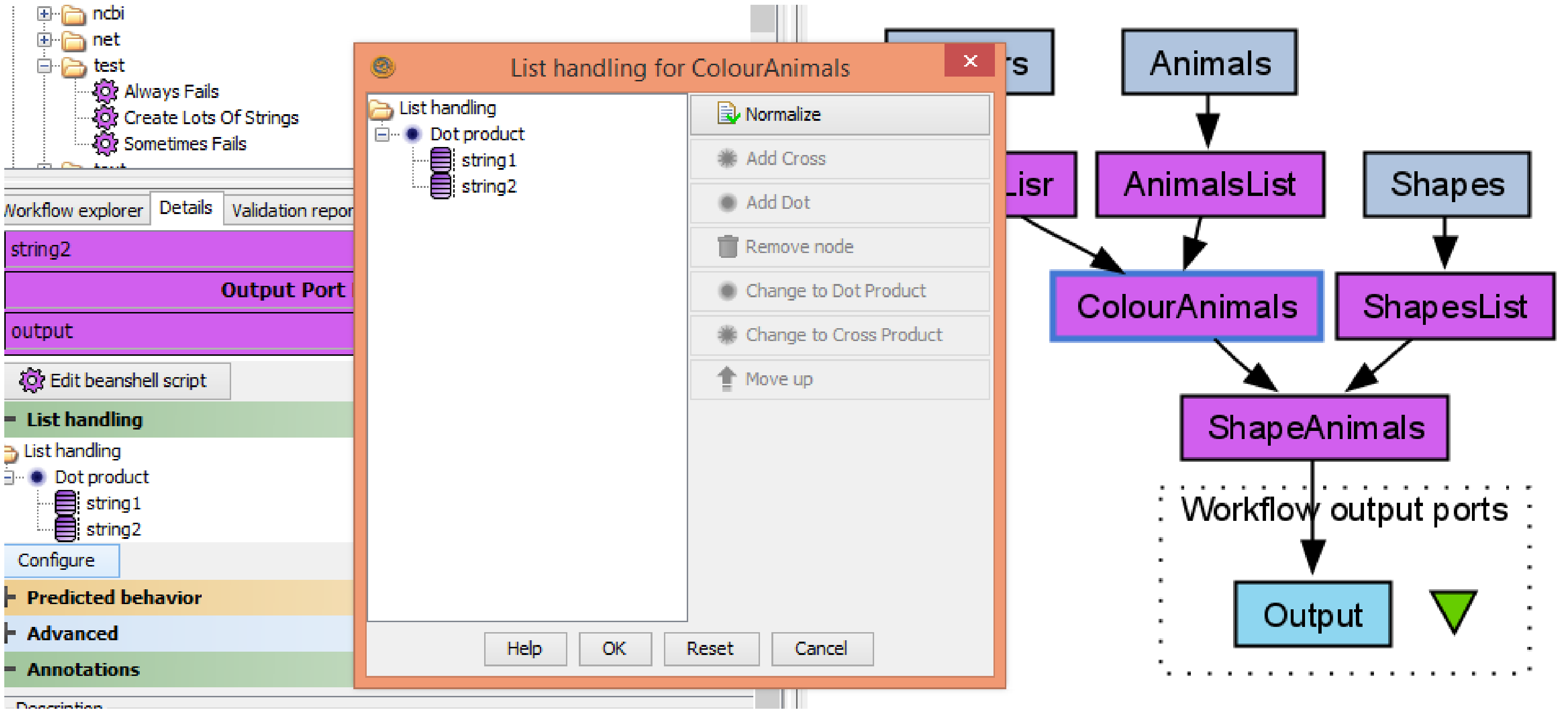
Exercise : List handling – example workflow

- Download and open the workflow "*Demonstration of configurable iteration*" from the workshop pack http://www.myexperiment.org/packs/641.html on myExperiment
- Read the workflow metadata to find out what the workflow does (by looking at the 'Details')
- Run the workflow and look at the results

Exercise : List handling - configuration

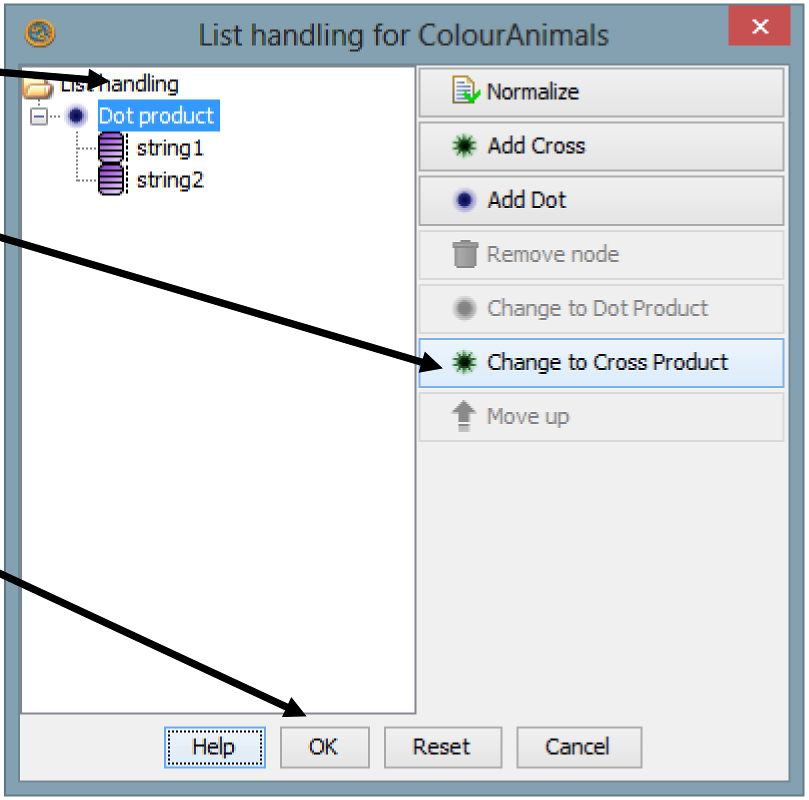
- Go back to the **Design** view
- Select the *ColourAnimals* service by clicking on it
- Select the **Details** tab in the workflow explorer, open **List handling** and click on **Configure**,
- or right-click on *ColourAnimals*, select **Configure running…** then **List handling…**
- Click on **Dot product** in the pop-up window. This allows you to switch to cross product (see the next slide)

Exercise : List handling – configuring - 1



Exercise : List handling – configuring - 2

- Click on **Dot Product**
- Click **Change to Cross Product** on the right
- Click **OK**
- Run the workflow again



Exercise : List handling - difference

- What is the difference between the results of the two runs? What does it mean to specify dot or cross product?

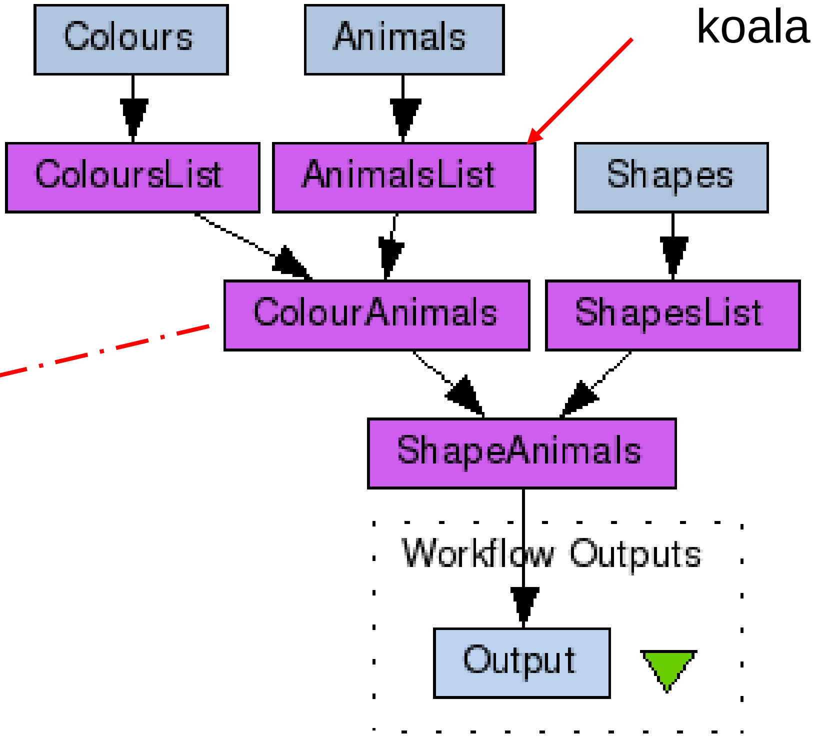
  NOTE: The iteration strategies are very important. Setting cross product instead of dot when you have 2000 data items can cause large and unnecessary increases in computation!

Exercise : List handling - workflow

e.g. red, green, blue, yellow

e.g. cat, donkey, koala

How does Taverna combine them?



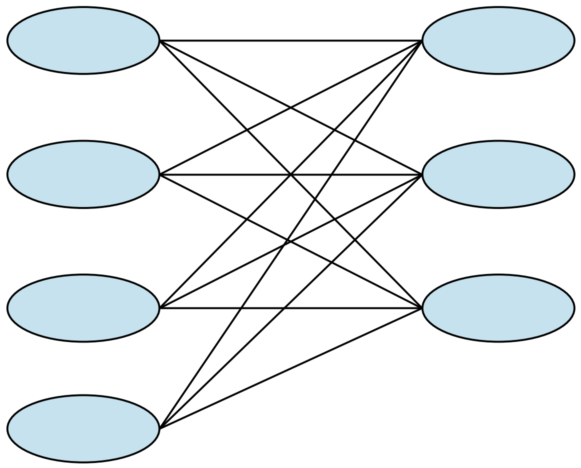
Exercise : List handling - Cross product

Red

Green

Blue

Yellow



Cat

Donkey

Koala

Red cat, red donkey, red koala

Green cat, green donkey, green koala

Blue cat, blue donkey, blue koala

Yellow cat, yellow donkey, yellow koala

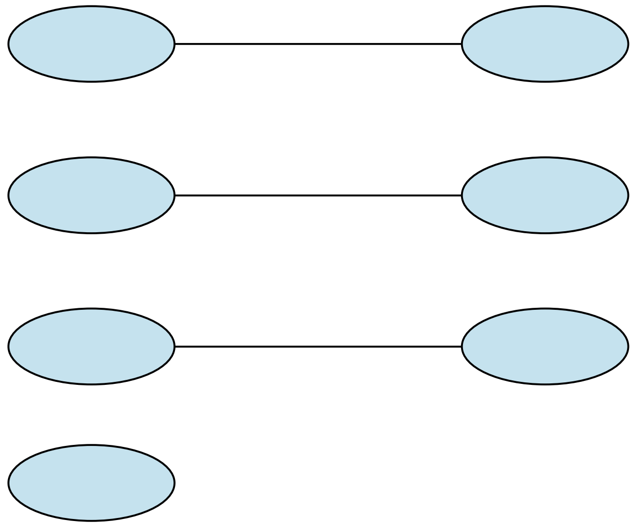
Exercise : List handling - Dot product

Red

Green

Blue

Yellow



Cat

Donkey

Koala

Red cat

Green donkey

Blue koala

There is no yellow animal because the list lengths don't match!

Exercise : List handling - summary

- The default in Taverna is cross product
- Be careful! All against all in large iterations give very big numbers!

Exercise : Looping - asynchronous

- Find the workflow "*EBI_InterproScan_broken*" in the workshop pack on myExperiment
- InterproScan analyses a given protein sequence (or set of sequences) for functional motifs and domains
- This workflow is asynchronous. This means that when you submit data to the *runInterproScan* service, it will return a jobID and place your job in a queue (this is very useful if your job will take a long time!)
- The *Status* nested workflow will query your job ID to find out if it is complete

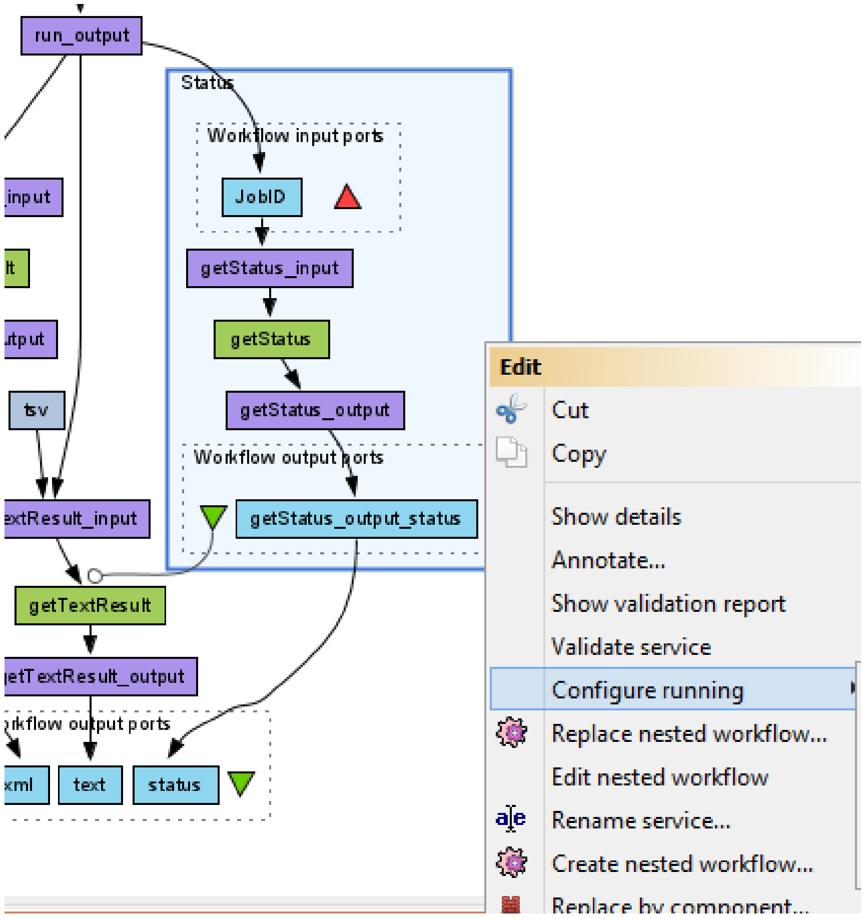
Exercise : Looping – no looping

- The default behaviour in a workflow is to call each service only once for each item of data – so what if your job has not finished when *Status* workflow asks?
- Download and run the workflow, using the default protein sequence and your own email address
- Almost every time, the workflow will fail because the results have not been returned before the workflow reaches the *get_results* service

Exercise : Looping - adding

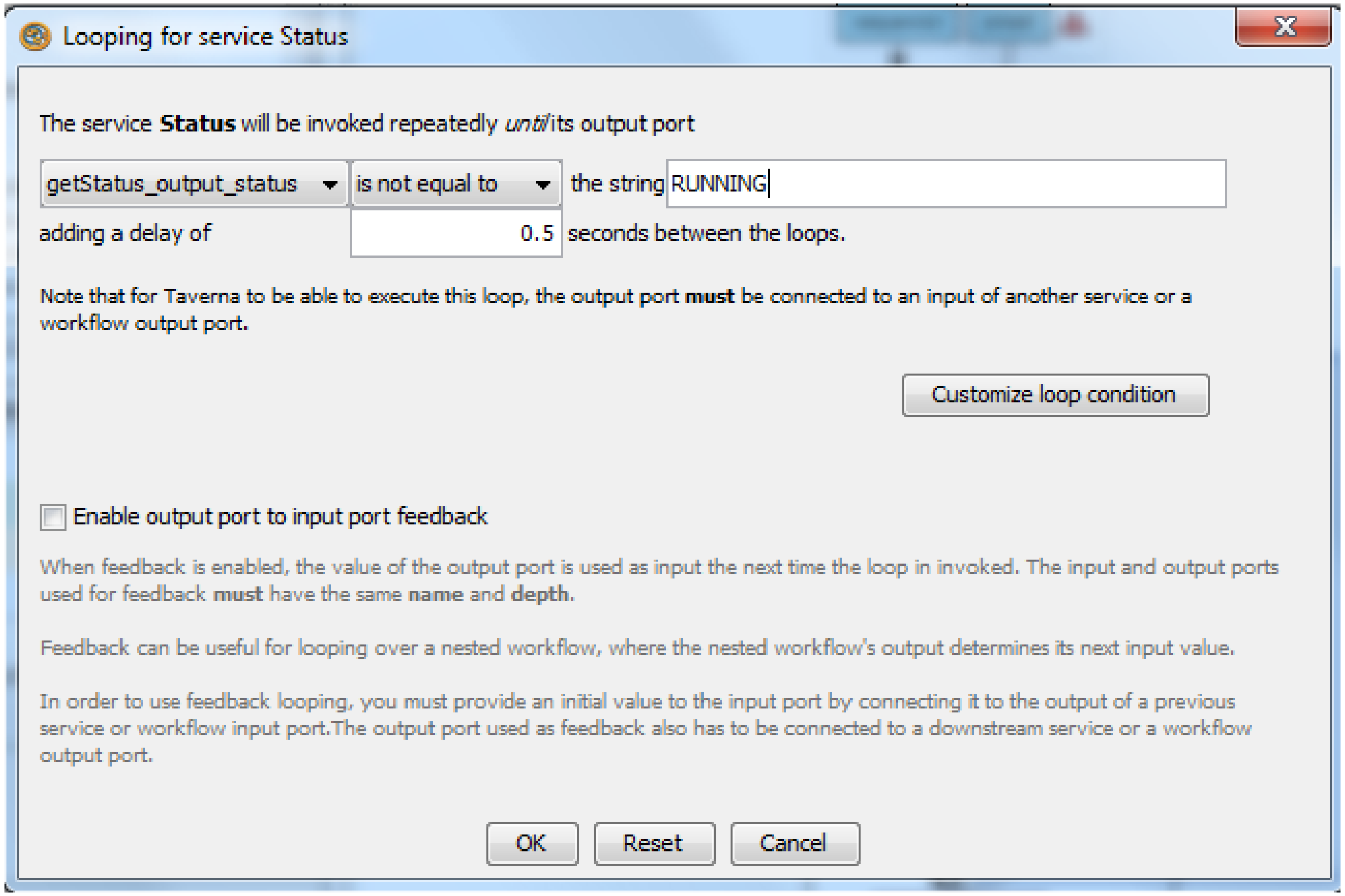
- This is where looping is useful. Taverna can keep running the *Status* service *until* it reports that the job is done.
- Go back to the **Design** view
- Select the *Status* nested workflow
- Select the **Details** tab in the workflow explorer, open **Advanced** and click on **Add looping**,
- or right-click on *Status*, select **Configure running…** then **Looping…**

Exercise : Looping - menu



Exercise : Looping - configuration

- Use the drop-down boxes in the looping window to set *getStatus_output_status* **is_not_equal_to** *RUNNING*



Looping for service Status

The service **Status** will be invoked repeatedly *until* its output port

getStatus_output_status | is not equal to | the string RUNNING

adding a delay of 0.5 seconds between the loops.

Note that for Taverna to be able to execute this loop, the output port **must** be connected to an input of another service or a workflow output port.

Customize loop condition

☐ Enable output port to input port feedback

When feedback is enabled, the value of the output port is used as input the next time the loop in invoked. The input and output ports used for feedback **must** have the same **name** and **depth**.

Feedback can be useful for looping over a nested workflow, where the nested workflow's output determines its next input value.

In order to use feedback looping, you must provide an initial value to the input port by connecting it to the output of a previous service or workflow input port.The output port used as feedback also has to be connected to a downstream service or a workflow output port.

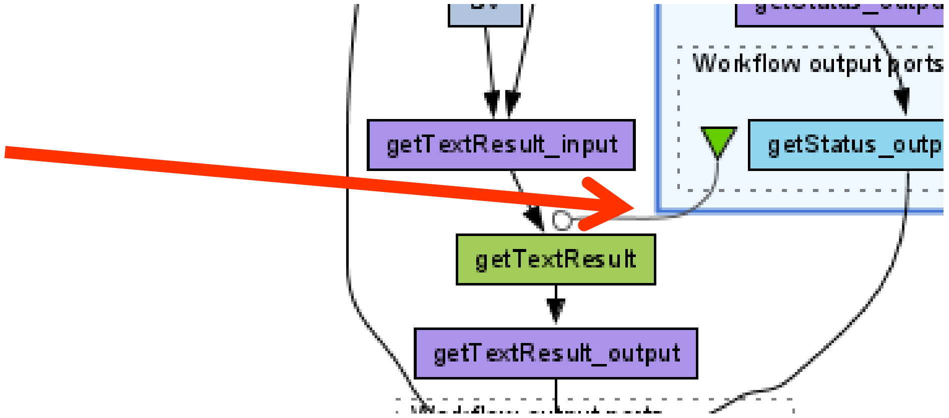
OK | Reset | Cancel

Exercise : Looping - configured

- Run the workflow again
- This time, the workflow will run until the *Status* nested workflow reports that it is either DONE, or it has an ERROR.
- You will see results for *text*, but you will still get an error for '*xml*'. This is because there is one more configuration to change – we also need **Control Links**

Exercise : Control Links - introduction

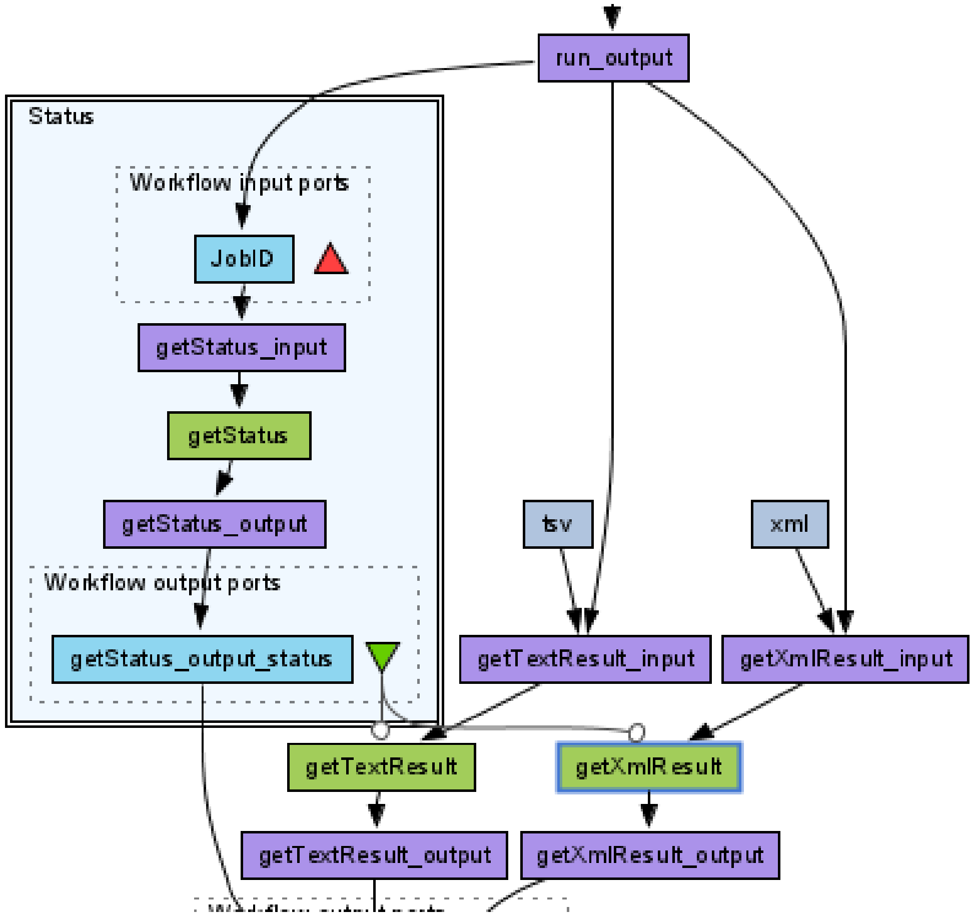
- A control link specifies that there is a dependency of one service on another even though there is no data flowing between them.
- A control link is a line with a white circle at the end that connects two services (see the link between the *Status* nested workflow and *getTextResult*)



Exercise : Control Links - adding

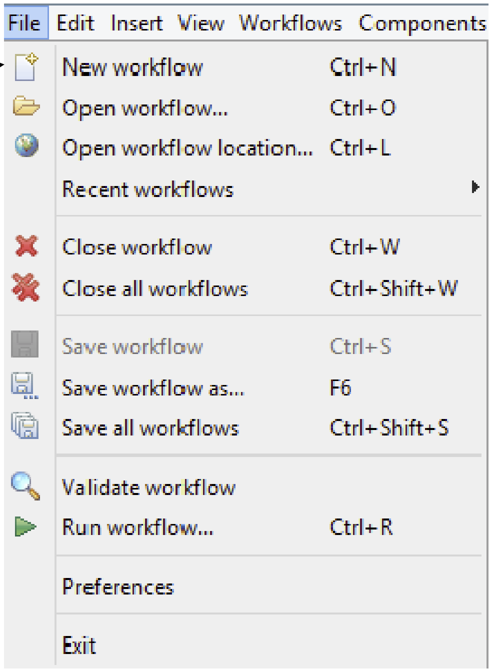
- We will add control links to the other output type
- Switch to the **Design** view
- Right-click on *getXmlResult* and select **Run after** from the drop down menu.
- Set it to **Run after** -> *Status*
- Save and run the workflow
- Now you will see each result returned

Exercise : Control Links - effect



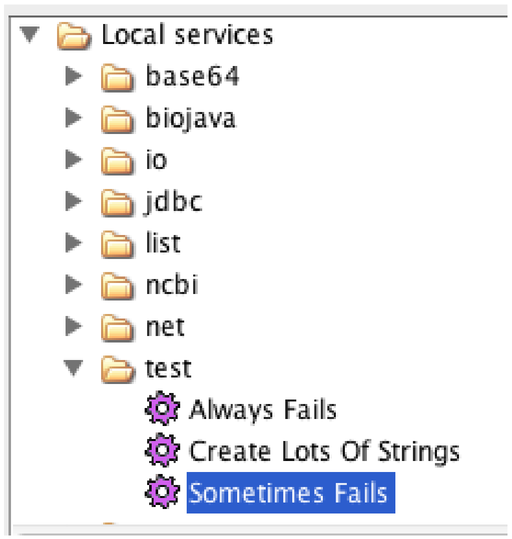
Exercise : Retries - Introduction

- Web services can sometimes fail due to network connectivity
- If you are iterating over lots of data items, you can guard against these temporary interruptions by adding retries to your workflow
- Create a **New workflow**



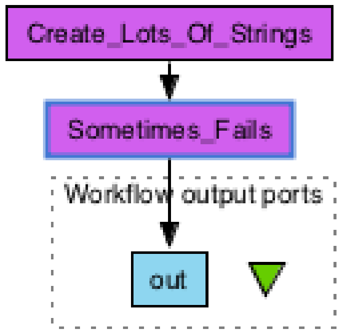
Exercise : Retries – Creating an example - 1

- In the **Service panel**,
- Select the service *Create Lots Of Strings* under **Available Services** -> **Local services** -> **test**
- Add it to the workflow by dragging it into the workflow diagram
- Also add *Sometimes Fails*



Exercise : Retries – Creating an example - 2

- Add an output port and connect the service as on the picture below
- Run the workflow as it is and count the number of failed iterations



Exercise : Retries - Configuring

- Now, select the *Sometimes_Fails* service and select the **Details** tab in the workflow explorer panel
- Click on **Advanced** and **Configure** for **Retry**
- In the pop-up box, change it so that it retries each service iteration 2 times
- Run the workflow again – how many failures do you get this time?
- Change the workflow to retry 5 times – does it work every time now?

Exercise : Parallel jobs

- If Taverna is iterating over lots of independent input data, you can improve the efficiency of the workflow by running those iterated jobs in parallel
- Run the Retries workflow again and time how long it takes
- Go back to the **Design** view, right-click on the *Sometimes_Fails* service, and select **Configure running…**
- This time select **Parallel jobs…** and change the maximum number to 20
- Run the workflow again
- Does it run faster?

Exercise : Parallel jobs - warning

- Setting parallel jobs makes your workflows run faster, but you should be careful if you are using remote services. Sometimes they have policies for the number of concurrent jobs individuals should run (e.g. The EBI ask that you do not submit more than 25 at once).
- If you exceed this number, your service invocations may be blocked by the provider. In extreme cases, the provider may block your whole institution!

Summary

- The tutorial has shown you how to use
  - list handling to change how Taverna matches up elements in a list,
  - looping to control when a service is “finished”,
  - control links to specify when a service can run
  - retries to handle failure and
  - parallel invocation to speed up workflow runs