Asynchronous services in Taverna

Stian Soiland-Reyes and Christian Brenninkmeijer
University of Manchester

Bonn University, 2014-09-01 / 2014-09-03

http://www.taverna.org.uk/
Most web services (REST and SOAP) are synchronous
- No response until the processing/results are ready

Problem:
- Long-running calculations (e.g. sequence alignment) can time out on the network
- Can’t tell if the service is taking a long time to process, or is simply failing to respond
Asynchronous services on the other hand works by multiple calls in a fixed pattern.

**Typical pattern:**
- `createJob` takes any inputs, and returns some kind of `jobId` – e.g. 001572
- `checkStatus` return the current `state` of the job – e.g. RUNNING
- `getResults` retrieve the outputs for the job
Although this is a common pattern, services vary in what they call each step and state. Typical names:

- createJob, submit, run, runAsync, launch, sequenceAlignment
  - Returning jobId, job, id, run

- checkStatus, job_state, check_job, isFinished
  - Might return more than just the status

- getResults, retrieveOutput, jobOut
  - Sometimes merged with checkStatus

- RUNNING, active, QUEUED, Finished, complete, ready

https://www.biocatalogue.org/services/3679/service_endpoint
A **control link** ensures `getResults` does not run until `checkStatus` is finished.

**Looping** on `checkStatus` to repeat until status is no longer `RUNNING`.

- Covering both `FINISHED` and `FAILED`.
- Might be a nested workflow.
- Wrapped as a **nested workflow** or **component** to appear as a single service within mother workflow.
“Looping” Exercise (in Advanced Taverna) details how to configure **looping** in a workflow using the EBI *InterproScan* protein signature recognition service.

“Control Links” Exercise details how to add **control links** to ensure results are not fetched before the job is complete.