An Introduction to Taverna Components

Stian Soiland-Reyes and Christian Brenninkmeijer
University of Manchester
materials by Aleksandra Pawlik

http://orcid.org/0000-0001-9842-9718
http://orcid.org/0000-0002-2937-7819
http://orcid.org/0000-0002-1279-5133
http://orcid.org/0000-0001-8418-6735

Bonn University, 2014-09-01
http://www.taverna.org.uk/
What is a component?

- Something that can be put into a workflow
  - Well described - what the component does
  - Behaves “well” - conforms to agreed good practice
  - Curated - someone looks after it
  - Produces and consumes data in agreed formats
  - Fails in described ways - meaningful error messages
  - Produces agreed type of provenance
- Documentation
- Example usage
Usefulness of components

- Hide complexity
- Predictable good behaviour
- Guaranteed to work together
- Can (in theory) check that data in a run conforms to the component specification
What is the agreement?

- The agreement is a condition of being in a “component family”
- Different domains, or even different uses within a domain, have different agreements
  - Astronomical data is not in the same formats as biodiversity data
  - Digital library components do not do the same tasks as biodiversity components
- Agreement is formalized as a “component profile”
• A component family is
  • a pack on myExperiment, or
  • a directory on your local machine
• A component is defined by a workflow (in a pack) in a component family pack
• Components are versioned by the myExperiment’s versioning
• Semantic annotations are stored in RDF as part of the workflow definition
• Collated semantics, including workflow structure, are combined on myExperiment.
Implementation
Component pack

- Contains:
  - Workflow ‘realizing’ the component
  - Example data
  - Documentation
  - Dependency specification
• A component family is shown in the service panel of Taverna workbench

• Components can be included within a Taverna workflow

• Components are **not** simply the same as nested workflows
  • You could think of them as nested workflows that obey a set of rules and where you cannot see what is nested (and should not care)
Components are created by annotating a workflow

- Choice of a component family and so profile
- Semantic annotation from the specified ontologies
- Validation against the profile
- Component saved into the component family

Can annotate:

- Workflow
- Input/Output ports
- Services inside workflow

Extensions to myExperiment for

- Pack snapshots
- Semantic collation
- Semantic searching
Semantic annotation

**Semantic Annotations**
- Annotation type: handlesMimetype
  - image/jp2
    - Change
    - Delete

- Annotation type: fits
  - Characterisation
    - Change
    - Delete

**Turtle annotations**
```
<>
  <http://purl.org/DP/components#fits>  
  <http://purl.org/DP/components#Characterisation> ;
  <http://purl.org/DP/components#handlesMimetype>  
  "image/jp2"^^<http://www.w3.org/2001/XMLSchema#string> .
```
Effect on workflows

- Use of components will allow
  - Component developers to work on the component
  - Component users to upgrade (or revert) the component versions
  - A workflow to remain ‘unchanged’ (if the component interfaces remain the same)
    - Powerful and dangerous
  - Proxies for components (re-run and re-play)

- Components are “black boxes” in the workflow and workflow runs
The workflow to call EBI InterproScan was quite complex.

It would be nice to be able to package that workflow up and be able to use it as a single service in other workflows.

That is exactly what components allow.
Components are grouped into component families

Component families are held in a component registry

myExperiment is a component registry

You can import a component family into the Service Panel

Click Import new services and then

Component service...
Selecting a component family

- In the dialog
- Select *myExperiment* component registry, and
- *Test components* family
- Click OK
Added component family

- In the **Service panel** you can now expand and see the *Test components* family
Adding a component to a workflow

- Create a new workflow
- Add the EBI_InterproScan component into the workflow
- Create input and output workflow ports and connect them to the ports of the component
EBI InterproScan component

Workflow input ports
- email
- sequence

Workflow output ports
- status
- text
- xml
Running the workflow

- You can now run the workflow
- The value for the sequence should be something like:

```
>sp|Q9BTV4|TMM43_HUMAN Transmembrane protein 43 OS=Homo sapiens GN=TMEM43 PE=1 SV=1
MAANYSSTSTRREHVKVTSSQPGFLERLSETSGGMFVGLMAFLLSFYLFNEGRALKT
ATSLAEGLSLVSPDSISHVAPENEGRLVHIIGALRTSKLLSDPNYGVHLPAVKLRRHVE
MYQWVETEESREYTEDGQVKKETRYSYNTEDREIINSKNDREIGHKNPSAMAVESFMA
TAPFVQIGRFFEGLIDKVDNFKSLSLKLEDPHVDIIRRGDFYYHSENPKYPEVGDLR
VSFSYAGLSGDDPDGLPAAHVVTIARQRGDLQLVPFSTKSGDTRAINLHHGDFSAEEVFHRE
LRSNSMKTWGLRAAGWMAFMGLNLNLTRLTYTLYIDWFPVFRDLVNIGLKAFAFCVATSLT
LLTVAAGWLFYRPLWALLIAGLALVPILVARTRVPAKKLE
```
Connecting components

- The workflow just contains the single service, we need to connect the component with other services
- In the **Design view**, delete the *sequence* workflow input port
  - Right click and select **Delete workflow input port**
- Add **Local Services** -> **ncbi** -> **Get Protein FASTA** to the workflow
- Connect the *outputText* of **Get Protein FASTA** to the *sequence* port of the **EBI_InterproScan**
- Connect the *id* port of **Get Protein FASTA** to a workflow input port
Your workflow should now look like:
- Run the workflow again
- You can use Q9BTV4 as the value for *id*
Is it really the complex workflow?

- In the **Results view** you can click on **Progress report**
- Expand **EBI_InterproScan**
- You can see all the services “hidden” inside the component
The menu has a “Components” option
Select “Create family”
In the pop-up window set the registry to local
Select a Profile (or see next slide if no profile available)
Enter the family name ("ProcessString")
Adding a Profile (if required)

- Find your local registry directory
  - Hint: Components/ Manage Registries
    - Registry Location

- In MyExperiment find the Empty profile
  - Hint: http://www.myexperiment.org/files/1027.html

- Down File into the local registry directory
Add a local service “Split string into string list by regular expression” (from ‘text’)

Add the input port and set the regular expression to space

Add a local service “Remove string duplicates” (from ‘list’)

Connect the output from “Split string into string list by regular expression” with the input of “Remove string duplicates”

Add a local service “Merge String List to a String” and connect its input with the “Remove string duplicates” output and set the separator to be a space
Taverna Components in practice
Select “Create component” from the “Components” menu

Provide a name for the component (Remove duplicates)

You should see a pink ribbon at the top

Save the component. You will see a warning message – it pops up because the component is not annotated. We can annotate it in the component details.
Using your Component

- Close any open workflows
- Add the component(s) to the service panel
  - Hint: Import Service/ Component Family
- Component registry: Local registry
- Component family: ProcessString
- Add the component to the workflow
  - Hint: Available services/ Components …
- Add input and output ports
- Run