Activities – Taverna 2 and 3

Taverna 2
Taverna 3
Rationale

- Separation of engine (execution platform) from UI
- Identification of types of activity
  - Needed for execution platform capabilities
- Formalized description of configuration (JSON schema)
  - No more xstream
Activity type and configuration

- Activities had a specific class of configuration
  - BeanshellActivityConfigurationBean

- Activities are of a type that is identified by a URI

- All configurations are JSON nodes
Activity configuration

- JSON is defined by a JSON schema

```json
{
  "$schema": "http://json-schema.org/draft-03/schema#",
  "title": "Beanshell activity configuration",
  "type": "object",
  "properties": {
    "@context": {
      "description": "JSON-LD context for interpreting the configuration as RDF",
      "required": true,
      "enum": ["http://ns.taverna.org.uk/2010/activity/beanshell.context.json"]
    },
    "script": {
      "title": "Beanshell Script",
      "description": "The beanshell script to be executed",
      "type": "string",
      "required": true,
      "default": ""
    }
  }
}
```
Activity configuration - cont

```json
{
  "$schema": "http://json-schema.org/draft-03/schema#",
  "title": "Dependency activity configuration",
  "type": "object",
  "properties": {
    "@context": {
      "description": "JSON-LD context for interpreting the configuration as RDF",
      "required": true,
      "enum": ["http://ns.taverna.org.uk/2010/activity/dependency.context.json"]
    },
    "classLoaderSharing": {
      "title": "ClassLoader Sharing Policy",
      "description": "Policy for sharing class loaders across multiple beanshell activities",
      "default": "workflow",
      "enum": ["workflow", "system"]
    },
    "localDependency": {
      "title": "Local Dependencies",
      "description": "Location of jar files to be added to the beanshell class loader",
      "type": "array",
      "items": { "type": "string" }
    }
  }
}
```
Activity ports

- Activities could derive their ports from external information e.g. WSDL
- Activity ports are now always specified in the workflow bundle
  - Speeds loading of workflows and potential offline editing
  - Consistency checking needed to ensure up-to-date
Activity execution

- Almost identical in Taverna 2 and Taverna 3
- Only differences should be JSON node v special class
Activity configuration

- ActivityConfigurationDialog just set the configuration bean
- ActivityConfigurationDialog now
  - sets the configuration JSON node
  - creates the activity ports of the activity
  - maps them to the ports of the processor
Service insertion

- Adding a service into a workflow configured the activity
- The activity determined the activity’s ports
- Adding a service into a workflow now asks the ServiceRegistry for the activity’s ports
Service Registry

- Adding a service into a workflow configured the activity
- The activity determined the activity’s ports
- Adding a service into a workflow now asks a ServiceRegistry for the activity’s ports
- ServiceRegistry is new
- Not to be confused with ServiceDescriptionRegistry
Service Registry - cont

- ServiceRegistry knows about a set of types of activity
  public Set<URI> getActivityTypes();
- The URI is the same one as in the Activity

- Can get JSON describing the JSON schema for the activity type
  public JsonNode getActivityConfigurationSchema(URI activityType)
Service Registry - cont

- ServiceRegistry can calculate the ports for an activity type given a configuration

  public Set<InputActivityPort> getActivityInputPorts
      (URI activityType,
       JsonNode configuration)

  public Set<OutputActivityPort> getActivityOutputPorts
      (URI activityType,
       JsonNode configuration)
ActivityFactory

- ServiceRegistry uses ActivityService and ActivityFactorys to determine the ports etc.

- Separation of the workflow-relevant parts of the Activity from the execution part

- Currently being updated