SCAPE

Scalable Preservation Environments

Workflow modelling environment

Authors

Donal Fellows (University of Manchester)

January 2014

This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).

Table of Contents

1 Introduction

The SCAPE project has selected Taverna as the workflow system for describing preservation actions to be enacted. Taverna workflows consist (in the version 2 series of software releases) of XML documents (typically stored in files with a *.t2flow* filename extension) that describe the contents of the workflow, in particular the processing elements in the workflow and the links between them that direct the flow of data. Though there are many ways in which an XML document may be produced, we provide a Workflow Design tool — the Taverna Workbench, see Section 2 — that allows the creation of workflows using a drag-and-drop graphical layout system.

We also provide an execution engine for Taverna workflows, see Section 3. Though the core of this execution engine is incorporated into the Workbench so that users may try out their workflows to see if they perform as expected, the Workbench is known to not scale up to very large executions due to the need to provide live feedback of the progress of executions via a GUI. We therefore also provide Taverna Server, a specialized execution engine that clients can access via a web-service interface. Taverna Server does not provide any graphical interface, instead focusing on providing a clean interface for running user-supplied workflows. We are enhancing Taverna Server with support for execution of Preservation Action Plans, as defined in WP14; see Section 3.2 for more details.

2 Workflow-modelling Environment

The Taverna Workbench 2.5 is a stand-alone Java application. It consists of two principal aspects, the workflow design perspective, and the workflow results perspective.

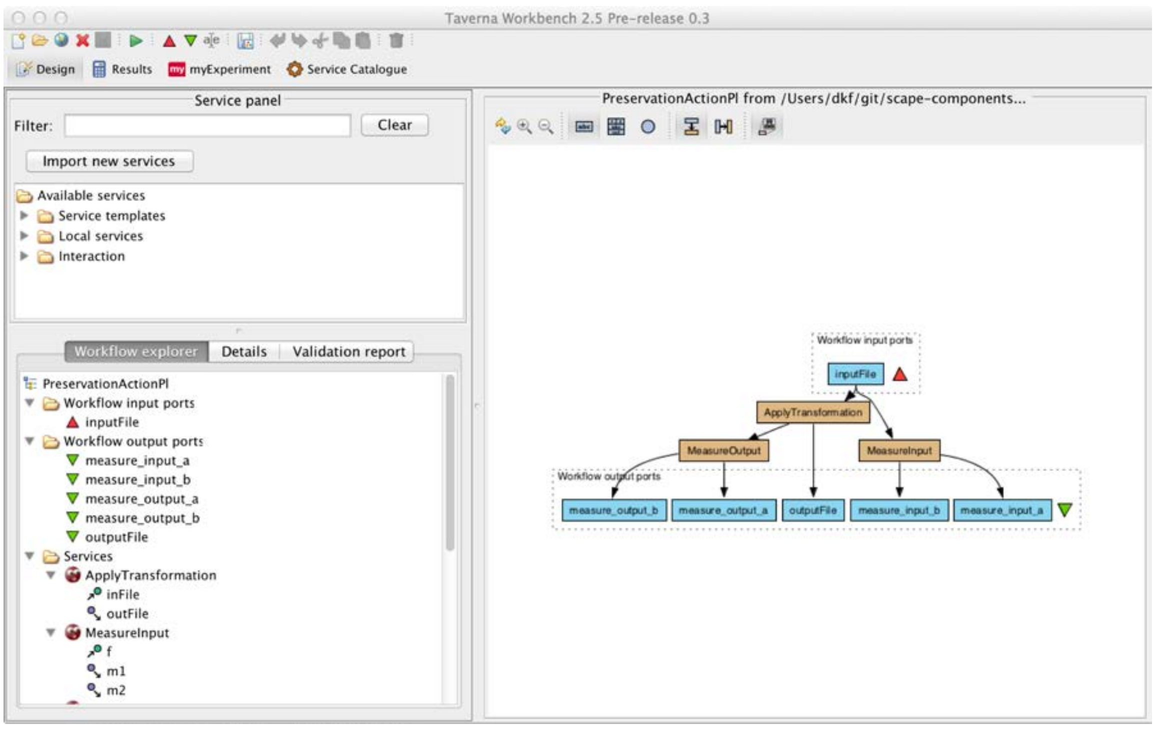
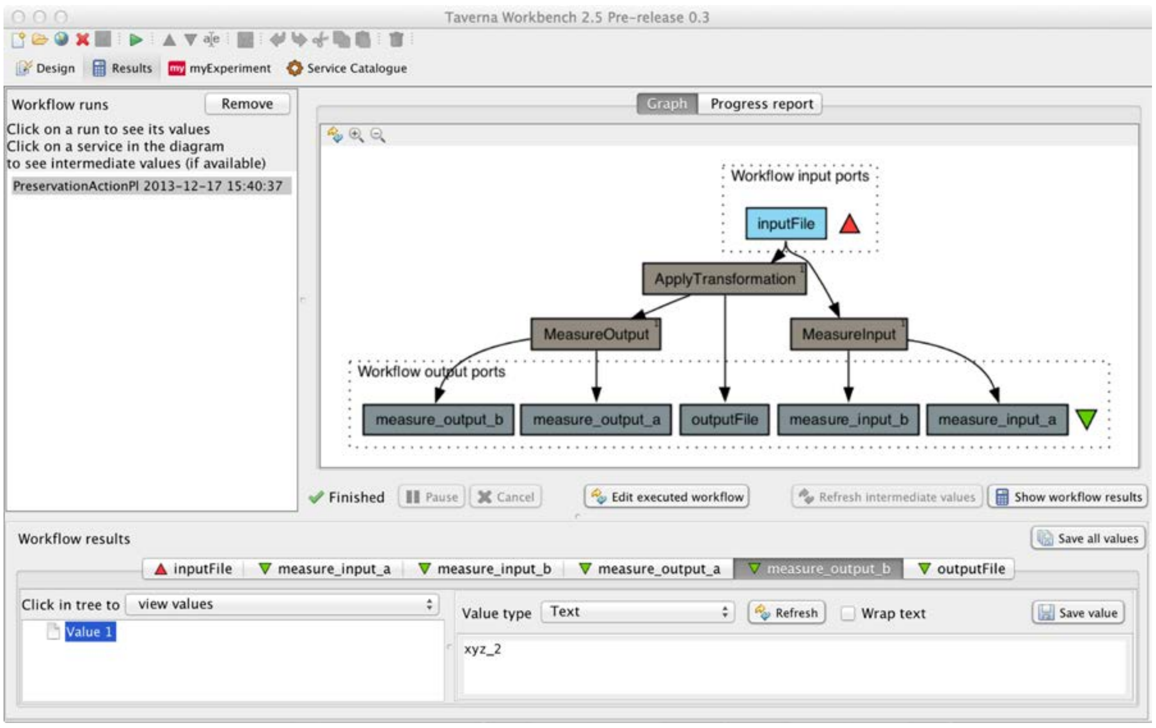


Figure 1: Taverna Workbench Design Perspective

The design perspective (see Figure 1) comprises a graph layout panel on the right side of the screen, and a stacked pair of panels on the left side of the screen: the service panel and the exploration/details panel. The graph layout panel provides the main user view of the workflow, allowing for direct manipulation of the elements of the workflow (i.e., the processors, the input ports, the output ports, and the links between them). The exploration/details panel provides an alternate tree-structured view of the workflow on its execution tab, while the details tab allows a user to find out more information about the particular configuration of a workflow element as well as giving the ability to configure advanced features of that element (e.g., how many times it may be executed in parallel); the final tab of that panel provides a validation report, allowing the user to find problems in their workflow prior to running it.



Click on a run to see its values
Click on a service in the diagram
to see intermediate values (if available)

PreservationActionPl 2013-12-17 15:40:37

xyz_2

Figure 2: Taverna Workbench Results Perspective

The results perspective (see Figure 2) is the other main view in Taverna Workbench, and is used when a running workflow is being viewed or the results of a workflow run are being studied. There are three main areas. The top-left is a panel used to select which workflow run is being viewed, the top right panel shows the workflow run in question (either graphically or in tabular form), and the bottom panel allows examination of the inputs and outputs, either of the overall workflow or (if provenance capture is enabled) of a processor in the workflow. For compound values (e.g., lists or lists of lists) the bottom panel is divided into two parts, the left allowing the selection of which value in the compound is to be studied, and the right allowing viewing of the specific leaf value. For simple leaf values, the right side is expanded to occupy the whole of the bottom panel.

2.1 Features

The key feature of the Taverna Workbench 2.5 over previous releases is the integrated support for creating and working with Components. Components are reusable user-defined high-level processors for use in workflows, where the components are defined in terms of a sub-workflow that is held in a searchable repository.

The other major feature of Taverna Workbench 2.5 is the interaction service, but this was not developed by SCAPE and is not used in SCAPE workflows.

Aside from these major features, there have also been a number of minor improvements to enhance the stability and scalability of the workbench, several new processor service types have been added (e.g., WebDAV support) and a number of processor service types (notably the Biomart, Biomoby and Soaplab service providers) have been removed on the grounds that they are expensive and very specific to a particular community (bioinformatics for genomics). It also includes the various SCAPE Component families in the service panel by default.

2.2 Availability

Taverna Workbench 2.5 is available for download from the Taverna website.
    http://www.taverna.org.uk/download/workbench/2-5/digital-preservation/
The user manual for Taverna Workbench is a separate document.
    http://dev.mygrid.org.uk/wiki/display/taverna/User+Manual
A guide on how to use and create components using the Taverna Workbench is online.
    http://www.slideshare.net/DonalFellows/scape-components-bootcamp
The source code for Taverna Workbench is on Google Code.
    https://code.google.com/p/taverna/source/browse/taverna/products/net.sf.taverna.t2.taverna-workbench/trunk

3 Execution Engine

Taverna Server 2.5.3 is an execution engine that takes user-supplied Taverna workflows and enacts them against supplied inputs, yielding the outputs of the workflow and providing descriptions of the activity that took place during the execution.

Deployments of Taverna Server 2.5.3 provide a number of service interfaces:

- Standard REST
- Standard SOAP
- Administrator REST
- Administrator HTML
- Administrator JMX

We are also in the process of extending Taverna Server to not just provide generic workflow enactment, but also to support specifically the kinds of workflows found in SCAPE. This is done through directly supporting an additional interface, the SCAPE Execution Service interface (a RESTful API), which allows Taverna Server to directly accept and enact SCAPE Preservation Action Plan documents. The support for this interface is on a custom branch of the source code of Taverna Server (the `scape-execution-interface` branch). This branch is derived from Taverna Server 2.5.3, and also supports *all* the features of Taverna Server 2.5.3.

Taverna Server supports this interface by transforming a Preservation Action Plan (supplied by PLATO via the Plan Management Service) into a fully enactable workflow. It then feeds that into the execution engine inside Taverna Server. We describe this in more detail in Section 3.2 below.

3.1 Features

Taverna Server 2.5.3 has a substantial number of features over Taverna Server 2.4 (the state of the service execution engine prior to the SCAPE project contributing effort). The key features are:

- Support for components
- Support for interactions (developed outside of SCAPE)
- Access to workflow logs
- Access to resource usage descriptions
- Access to provenance data

The features of the Taverna Server 2.5.3 distribution are discussed at more length in its user manual.

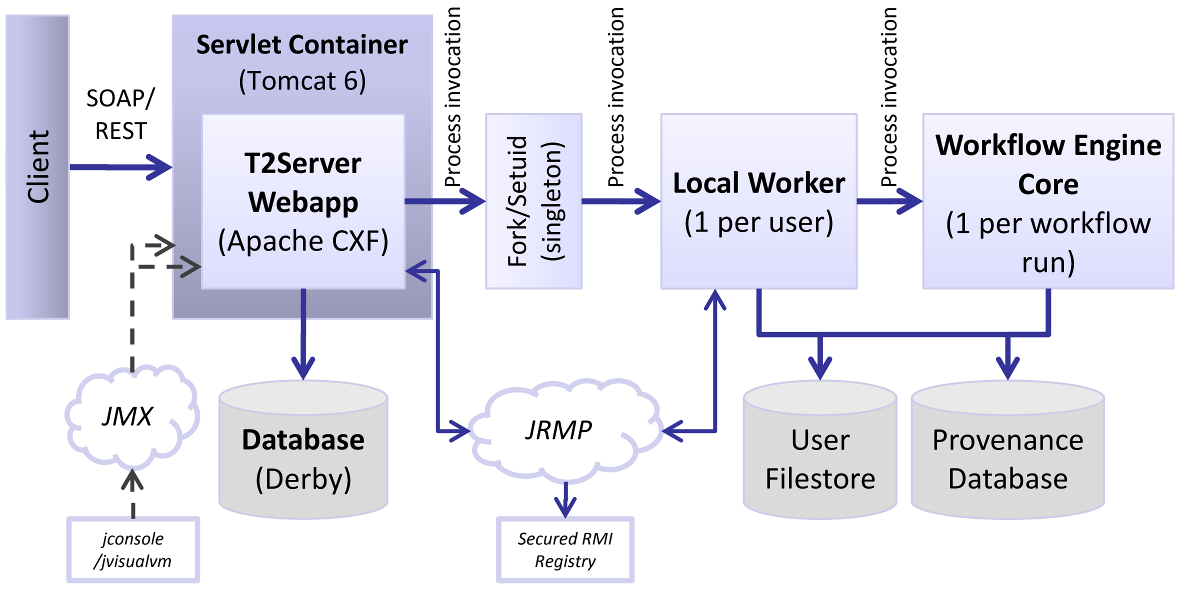


Figure 3: Summary of Architecture of Taverna Server

The architecture of Taverna Server 2.5.3 (see Figure 3) is substantially similar to that of previous versions of Taverna Server; the server overall consists of the T2Server webapp (the front-end engine), a secure fork engine, a local worker process that runs in a privilege-separated configuration as the user who submitted the workflow, and a workflow engine process that does the execution (so that any failures due to erroneous workflows only affect a single workflow run, not the whole service). The principal differences are in that the Workflow Engine Core has been enhanced with extra capabilities, the Local Worker exerts tighter control (and more customizable) over the resources used by the workflow engine core, and the provenance is now exposed as an explicit part of the workflow run results. These changes substantially enhance the scalability of the overall system, as well as allowing clients to discover more exactly what their code actually performed.

The other key change is that the build mechanism for Taverna Server has been substantially enhanced so as to reduce the overhead associated with starting a workflow run. This is done by pre-stressing the workflow engine core so that its internal caches can be constructed prior to packaging of the overall Taverna Server for distribution, rather than having those caches constructed upon invocation of each workflow run.

3.2 SCAPE Execution Service Extension Prototype

The prototype of Taverna Server for SCAPE supports the SCAPE Execution Service interface, allowing the submission of a Preservation Action Plan document (PAP, produced by PLATO and stored in the Plan Management Service; see D4.1 and WP14) to the service. The PAP contains within it a description of a collection of digital objects, a workflow describing the transformation operation (or operations) to apply to each of those digital objects, and optionally a definition of how to automatically assess whether the transformation was successful. The SCAPE Execution Service interface states that the PAP is submitted by POSTing it to a REST endpoint and that once the results of the processing are available, a notification is sent back to the Plan Management Service.

3.2.1 Architecture Summary

A Preservation Action Plan, submitted to Taverna Server's SCAPE Execution Service interface, is enacted as follows (see Figure 4):

1. Transforming the workflow in the plan into an executable form (see Section 3.2.2 below),
2. Instantiating that workflow as a workflow run object within the Taverna Server execution platform,
3. Arranging for the other parts of the preservation action plan to become workflow run inputs,
4. Setting the plan executing,
    a. This in turn will cause the digital objects to be retrieved from the repository,
    b. The transformations and analyses to be applied to them,
    c. The results to be assessed,
    d. Successful transformations to be staged back to the repository, and
    e. A notification of success or failure sent to the Plan Management Service. (This will cause the result report, together with any provenance traces and resource usage descriptions, to be transferred out of the Taverna Server installation.)
5. Finally, the workflow run is deleted, along with all associated resources, sometime after the run has finished executing. This happens either automatically after a timeout, or when the run is explicitly deleted.

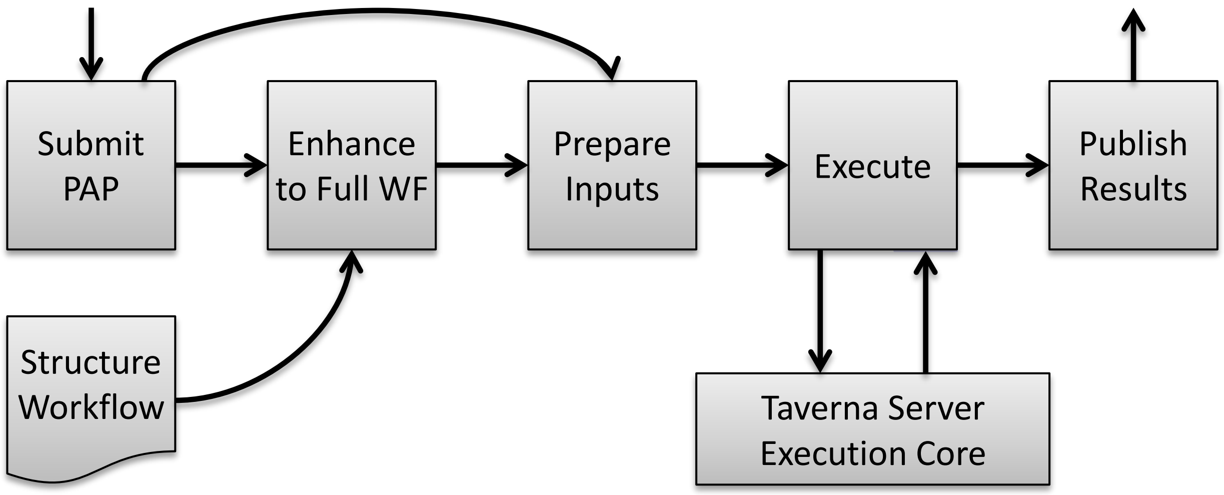


Figure 4: Overall Model of Execution of SCAPE Preservation Action Plans in Taverna Server

We plan to enhance Taverna Server with the PPL by using the PPL/Hadoop as an alternative execution strategy in step 4 above; the PPL will be applied to the transformed workflow and the result enacted on the Hadoop platform. From an external perspective, the use of the PPL/Hadoop will be transparent.

3.2.2 Transformation Engine

Because the workflow in a Preservation Action Plan only describes the transformations and measurements to be applied to a single digital object, it is necessary to transform the PAP workflow into a workflow that can be enacted. This Enactment Workflow has the PAP workflow as a sub-workflow of itself, and adds in the iteration over the overall specified collection of digital objects, the access to the repository holding the digital objects, and the preparation of a report describing whether each of the transformations was successful. Though it is a slight oversimplification, the PAP workflow is effectively the Map part of a logical MapReduce job, and the Enactment Workflow adds a common Reduce part.

To do this processing, the workflow within the PAP (see Figure 5) is injected into an outer workflow — the Structure Workflow (see Figure 7) — along with some extra utility components to form a new workflow, the Enactment Workflow (see Figure 8). This Enactment Workflow is then executed through the Taverna Server engine, using the non-workflow parts of the PAP as inputs to the workflow. It is a key assumption of this process that the results of applying a transformation to one digital object should in principle have no effect on the other digital objects to be transformed; though this is not going to be true for all possible transformations, this does describe a significant fraction of the transformations used in practice and makes the overall process complexity be capable in theory of linearly scaling with the number of objects to be transformed. Transformations that require all objects to be compared with all others are, in contrast, clearly going to be much harder to make scale using automatic techniques.

The execution interface is implemented as an extra REST endpoint supported by the T2Server Webapp part of Taverna Server (see Figure 3). The implementation of the execution interface invokes the transformation engine (described in more detail below) before injecting the result into the lower-level implementation engine for full enactment.

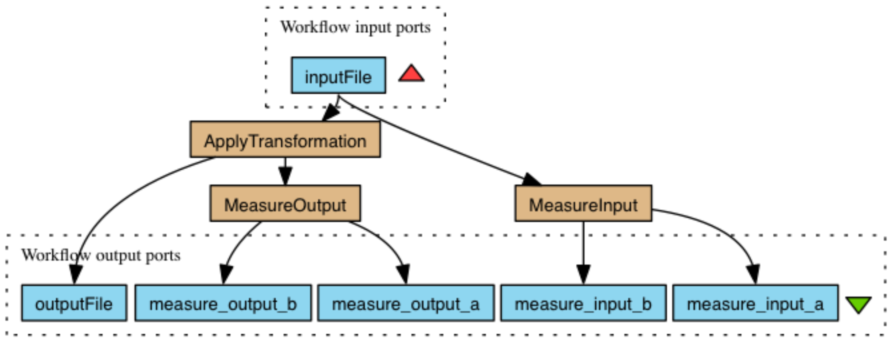


Figure 5: A simple Preservation Action Plan workflow example

The key feature of a PAP from the perspective of the execution injection system (a form of compilation transformation) is that it has one input port that describes the digital object that is being transformed, one output port that describes the entities on the file system (files or directories) that were produced by the transformation, and a number of other output ports that produce measures of the transformation. Those measures can be made via the application of SCAPE Characterisation Components (see WP9) to the input entities, or to the output entities, or to any intermediate entities produced within the PAP. They can also be made via the application of SCAPE QA Components to pairs of entities, when they will be measures relating to the comparisons and not to the individual entities compared. Each output port that carries a measure is annotated with information describing what metric was applied to get the measure, and with what it is a measure of (as otherwise there are metrics that are equally applicable to both input and output objects).

The other parts of a PAP are the collection of objects to which to apply the PAP workflow, and a rule (encoded as Schematron[1]) that describes whether a particular transformation was successful. An example document that might correspond to measures produced by the workflow in Figure 5 is given in Figure 6, with dummy values throughout (in particular, when used with a practical PAP the type field would be the name as in the SCAPE metric ontology, and the values would be the outputs of the relevant components in the PAP):

```
<measures>
    <measure subject="input" type="a">123</measure>
    <measure subject="input" type="b">234</measure>
    <measure subject="output" type="a">345</measure>
    <measure subject="output" type="b">456</measure>
    <measure subject="comparison" type="c">789</measure>
</measures>
```

Figure 6: Example of measures XML document, with dummy values

[1] ISO/IEC FDIS 19757-3: *Rule-based validation - Schematron*.

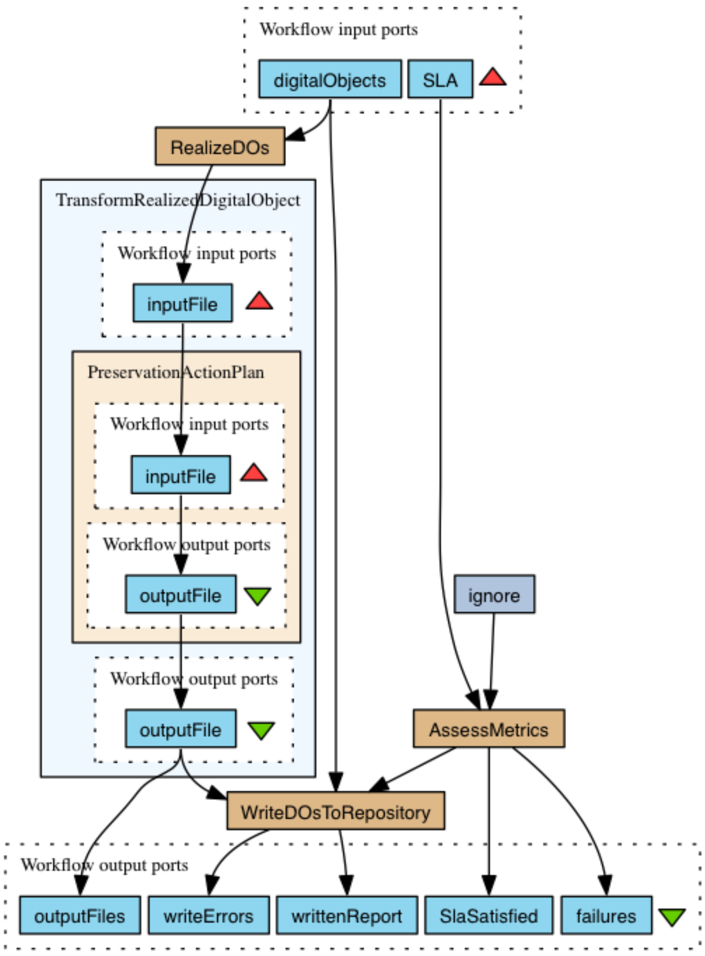


Figure 7: An example of a Structure Workflow

The Structure Workflow — intended to be managed by the administrator of the deployment of Taverna Server within which it is deployed — defines how the digital object references in the submitted PAP are retrieved from the configured repository, and then feeds the instantiated objects through an (initially-empty) inner workflow to perform the transformation. After that, it collects the combined measures before applying an assessment to the combined measures to determine whether the transformation was successful; that measure is provided as an input to the workflow as part of the overall PAP, and will in practice be a Schematron document. The final steps of the Structure Workflow are to stage the results of *successful* transformations to the repository[2], to produce a report describing the successful and failed transformations, and to notify the Plan Management Service (the instantiation of the Plan Management API, see D4.1) about the outcome of the transformation.

[2] Formally, the source and destination repositories do not need to be the same. This allows the content holding institution to review the processing of the PAP to determine for themselves whether they trust that the transformation did what they wanted it to and to the quality they desired.

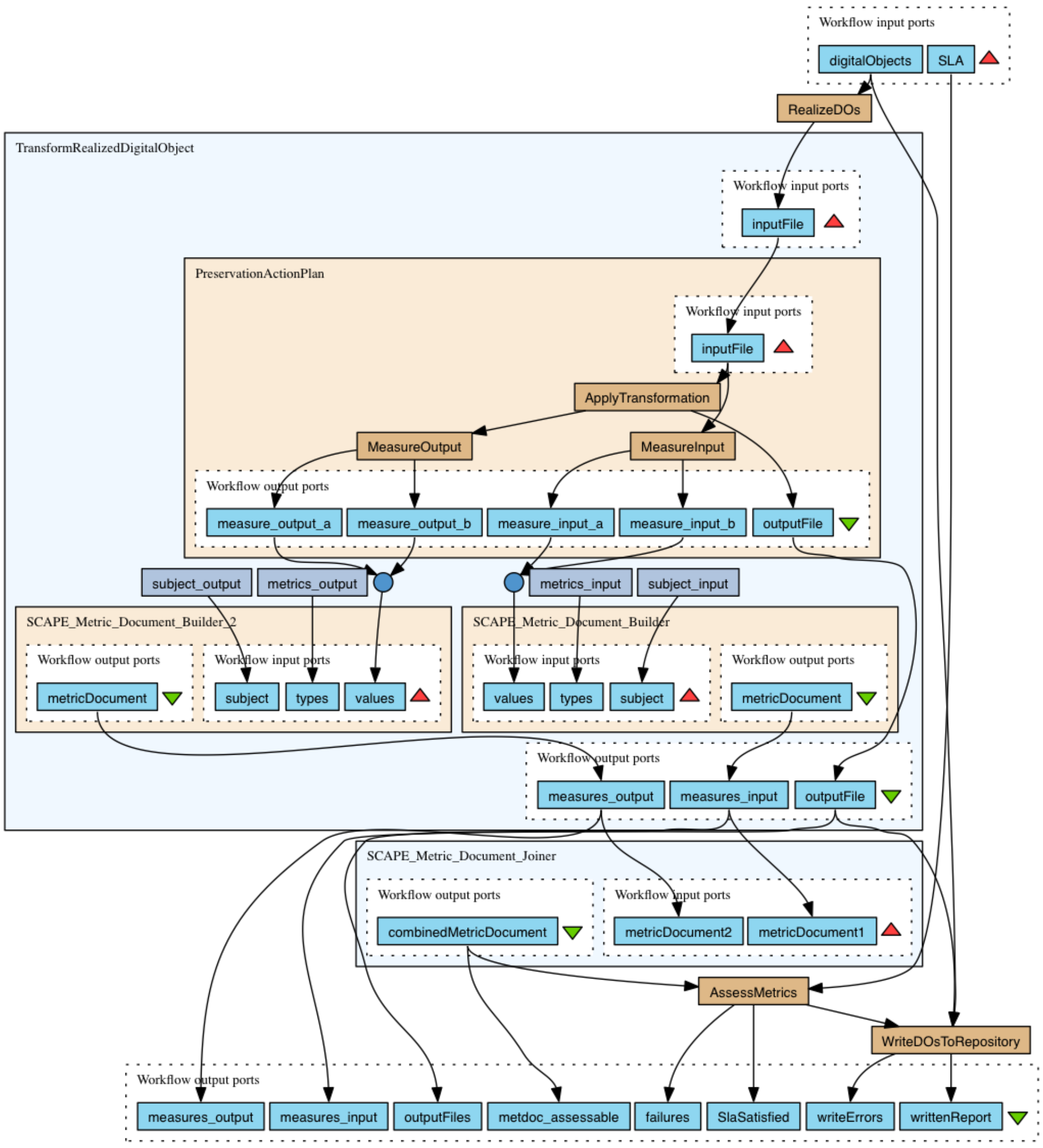


Figure 8: The Enactment Workflow generated by combining the PAP workflow and the structure workflow

The combined Enactment Workflow is formed within Taverna Server by the application of a mechanical transformation that works by plugging the PAP workflow into the innermost part of the structure workflow and then creating and connecting up appropriate utility components to cause the metadata encoded as annotations on PAP workflow ports to be manifested as concrete parts of a generated XML document that describes the entire results. It is this document that will have the Schematron assessment applied to it. The non-workflow parts of the PAP become instead arguments to the overall enactment workflow.

3.2.3 REST API Description

The REST interface supported by Taverna Server to allow access to the ability to enact Preservation Action Plans is exactly as described here. The `{serviceAddress}` parameter is a URL that describes the location of the service. The `{id}` parameter is an arbitrary string that does not contain a "/". All operations require authentication; Taverna Server supports authentication via HTTP Basic

Auth, and it is recommended that Taverna Server be deployed to use HTTPS to secure its communications with clients.

Query Current Preservation Action Plan Enactment Jobs

`GET {serviceAddress}`

Returns XML document describing the current Preservation Action Plans enactments being processed via this interface. The document will contain links pointing to the current job resources.

Accept New Preservation Action Plan

`POST {serviceAddress}`

Accepts a Preservation Action Plan and *asynchronously* instantiates a workflow run that enacts the plan. When the plan is syntactically correct, returns a redirect to the job resource that is used to track and manage the enactment.

Query Preservation Action Plan Enactment Information

`GET {serviceAddress}/{id}`

Returns an XML document describing the job that was created to enact a Preservation Action Plan. This includes a link to the resource that models the address that completion notifications should be pushed to, a link to the workflow document that was actually enacted (i.e., post transformation), a link to a resource that allows the execution status to be queried for (a standard feature of Taverna Server), and after execution has finished, a link to a WebDAV directory holding the output files, and a link to the workflow run provenance bundle.

Destroy Preservation Action Plan Enactment

`DELETE {serviceAddress}/{id}`

Destroys a particular enactment of the Preservation Action Plan and deletes all the resources associated with it. If the enactment is executing, this terminates the execution immediately.

Query Preservation Action Plan Completion Notification Address

`GET {serviceAddress}/{id}/notification`

Returns a plain text document containing the address that a notification of completion of a Preservation Action Plan enactment job will be pushed to.

Set Preservation Action Plan Completion Notification Address

`PUT {serviceAddress}/{id}/notification`

Sets (as plain text) the address that a notification of completion of a Preservation Action Plan enactment job will be pushed to. An empty document indicates that no notification should be sent.

3.2.4 Future Work: Enhancement with PPL

The Enactment Workflow has the property (for reasonable PAP workflows) that all the information combination operations in it are joins. Indeed, as noted above, the Enactment Workflow has a form that is already logically very close to that of a MapReduce job, and the operations applied to one digital object have principally no effect on other digital objects. This means that the enactment workflow will be comparatively straightforward for the PPL (being developed in WP6) to compile into an optimized form for enactment on a highly parallel non-Taverna platform. However, this is distinct from the transformation described above that converts the PAP workflow into the enactment workflow, as that is necessary for the PAP to be correctly enacted at all; the PAP does not describe the process of accessing the repository or of computing whether the transformation was correct.

Once a reasonably robust PPL is available, it will integrate into the execution flow model as in Figure 9. This should be seen as being very similar to the model used in Figure 4; only the implementation of the *Execute* stage is substantially different.

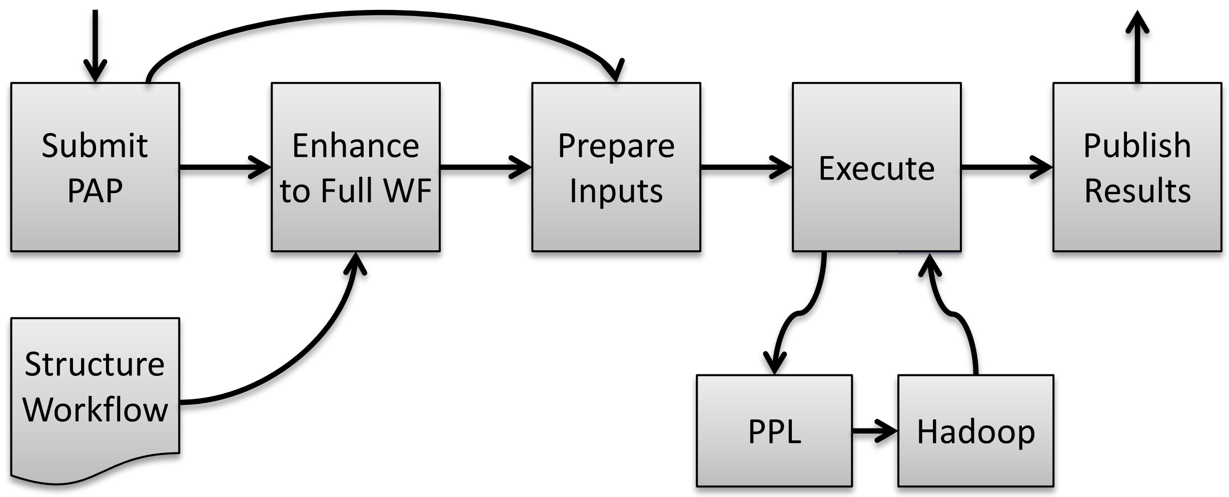


Figure 9: Overall Model of Execution of SCAPE Preservation Action Plans with PPL

3.3 Availability

The release of Taverna Server 2.5.3 is available for download.
https://launchpad.net/taverna-server/2.5.x/2.5.3
The installation guide is available as a separate document.
https://github.com/myGrid/taverna-server/blob/2.5.3/install.pdf?raw=true
The user manual is available as a separate document.
https://github.com/myGrid/taverna-server/blob/2.5.3/usage.pdf?raw=true
The source code for Taverna Server is on Github.
https://github.com/myGrid/taverna-server/tree/2.5.3

The build of Taverna Server with support for the SCAPE Execution Service model is currently a developer prototype only. The source code for this prototype is on Github:
https://github.com/myGrid/taverna-server/tree/scape-execution-interface

4 Glossary

Term	Definition
PAP	Abbreviation: *Preservation Action Plan*
Plan Management Service	A service that holds a *Preservation Plan* and manages its lifecycle. This is a service that implements the Plan Management API, described in Deliverable D4.1.
PLATO	A web-based tool that creates a *Preservation Plan* and provides a user interface for viewing, managing and updating that plan. The plan itself is stored in the *Plan Management Service* after creation.
PPL	Abbreviation: *Program for parallel Preservation Load*
Preservation Action Plan	A document that is part of a *Preservation Plan* defined by *PLATO* and stored in the *Plan Management Service*.

Term	Definition
Preservation Plan	An updateable document that acts as the central data model of a preservation planning activity. Stored in the *Plan Management Service*.
Program for parallel Preservation Load	A tool for converting a Taverna Workflow (using a restricted set of suitable operations) into a program that can be enacted efficiently on a Hadoop cluster. See work-package WP6.
SCAPE Characterisation Components	Characterisation components are a family of *SCAPE Components* (defined to wrap tools produced in WP9) that compute one or more properties of a *single* instantiated digital object or file. The output ports that produce measures are always annotated with the metric (in the *SCAPE Ontology*) that describes what the component computes.
SCAPE Components	SCAPE components are *Taverna Components*, identified by the SCAPE Preservation Components sub-project, that conform to the general SCAPE requirements for having annotation of their behaviour, inputs and outputs. SCAPE components may be stored in the SCAPE Component Catalogue, which is a part of the myExperiment web service.
SCAPE Migration Components	Migration components are a family of *SCAPE Components* (defined to wrap tools produced in WP10) that apply a transformation to an instantiated digital object or file to produce a new file. The input is annotated with a term (from the *SCAPE Ontology*) that says what sort of digital object/file is accepted, and the output is annotated with a term that says what sort of file is produced.
SCAPE Ontology	The SCAPE Ontology is an OWL ontology that formally defines the terms used by computing systems in SCAPE.
SCAPE QA Components	QA components are a family of *SCAPE Components* (defined to wrap tools produced in WP11) that compute a comparison between *two* instantiated digital objects or two files. They produce at least one output that has a measure of similarity between the inputs, and that output is annotated with the metric (in the *SCAPE Ontology*) that describes the nature of the similarity metric.
SCAPE Utility Components	Utility components are a family of *Taverna Components* that provide miscellaneous capabilities required for constructing SCAPE workflows, but which are not a core feature of the SCAPE preservation planning process. For example, they can provide assembly and manipulation of XML documents that contain collections of measures of workflows.

Term	Definition
Taverna Components	Taverna components are *Taverna workflow* fragments that are stored independently of the workflows that they are used in, and that are semantically annotated with information about what the behaviour of the workflow fragment is. They are logically related to a programming language shared library, though the mechanisms involved differ. Taverna components are stored in a component repository. This repository can either be a local directory, or a remote service that supports the Taverna Component API[3]; the Taverna team supports myExperiment as such a service. Only components that are stored in a publically accessible service can be used by a *Taverna workflow* that has been sent to a system that was not originally used to create it. This will be described in depth in Deliverable D7.3.
Taverna Server	Taverna Server is a multi-user service that can execute *Taverna workflows*. Clients do not need to understand those workflows in order to execute them.
Taverna Workbench	The Taverna Workbench is a desktop application for creating, editing and executing *Taverna workflows*.
Taverna workflow	A Taverna workflow is a parallel data-processing program that can be executed by *Taverna Workbench* or *Taverna Server*. It is stored as an XML file, and has a graphical rendering.

[3] http://wiki.myexperiment.org/index.php/Developer:Components