THE UNIVERSITY OF SHEFFIELD

CENTER FOR COMPUTATION IMAGING AND SIMULATION TECHNOLOGIES IN BIOMEDICINE

SCIENTIFIC SOFTWARE DEVELOPMENT TEAM

TECHNICAL REPORT

WORKFLOWS COMPOSITION, INTEGRATION AND EXECUTION ON VPH-SHARE's WP6

ABSTRACT

This document describes CISTIB's contribution on the development and integration of workflows services and tools that enable VPH-Share users to develop, test and validate complex workflows, also taking advantage of other key components of the VPH-Infostructure: The Master Interface, the Cloud Manager and the Data services.

Contents

LIST OF FIGURES

1 Introduction

The User Access System is the main entry point for researchers and clinicians willing to access the VPH-Share infrastructure services. As such it should contain tools that enrich the user's experience and facilitate the interaction with the platform, exposing in an intuitive way all the data and services that constitute the infrastructure.

During the VPH-Share project several tools have been developed to compose, deploy and execute biomedical workflows. In this document, we provide a detailed description of these tools and functionalities, which constitute the production prototype of the workflow composition and execution tools that have been developed by the WP6 of the VPH-Share project.

Most of the work has been done behind the scenes to provide the user with friendly interfaces and reliable tools that are able to run smoothly through the VPH-Share infostructure. In particular, the current prototype includes the main facilities that allow the user to start using the infrastructure and implement basic use case scenarios.

Work Package 6 (WP6) is an integration work package and therefore some of the interactions with the other project Work Packages (WP) are also reflected in this document. Unfortunately, full details of these other WPs are not included here, see http://vph-share.eu/content/documents instead. Nevertheless, we encourage the reader to also review the "D6.5: Production Deployment of User Access Systems" document, which contains extended information about the User Access Systems.

The following sections describe the developed tools in detail:

- Architecture of the workflow services and management
- GIMAS WebServices plugin to make command line tools available as web services
- Specification of services requiring user interaction
- VPH-Share plugin
    - Desktop composition and execution tool
    - Web composition and execution through Taverna Online
    - Support for workflows with long execution times
- Workflow execution through the MI
- Batch execution
    - Desktop batch execution
    - Web batch execution
- Workflow Manager API
- Data Provenance and Semantic
- Workflow service monitoring

2 ARCHITECTURE OF THE WORKFLOW SERVICES AND MANAGEMENT

This section provides a brief review of the architecture, leaving the more detailed description for the following sections.

The final architecture for workflow composition, integration and execution is composed of client-side and server-side components, see Figure 1.

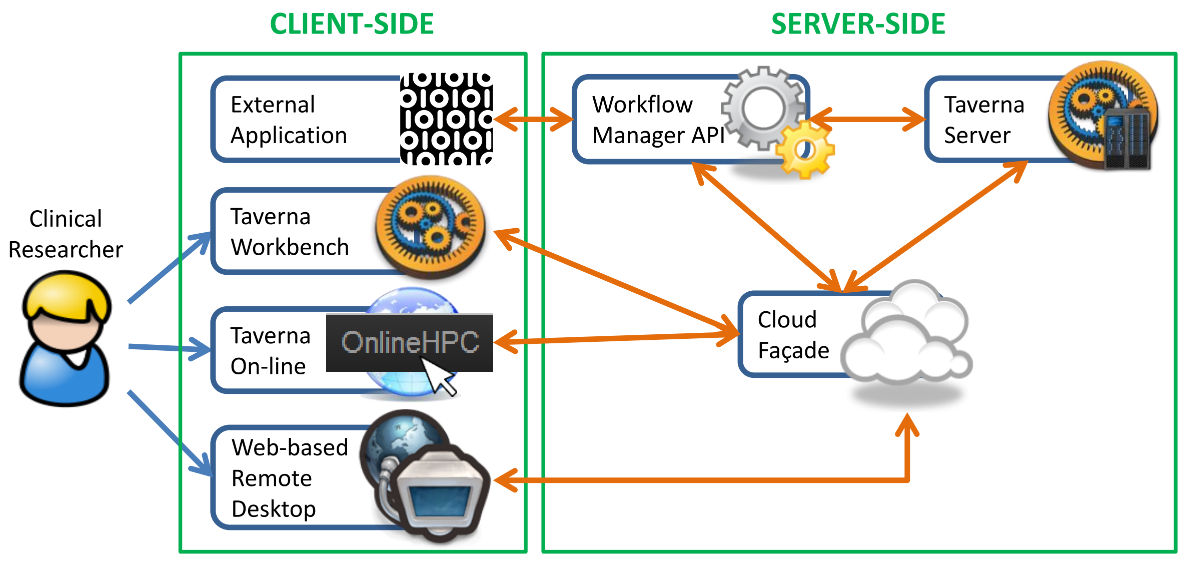


Figure 1. Workflow Management Architecture overview.

On the client-side, a biomedical workflow can be composed and executed by a clinical researcher using two different applications: Taverna Workbench and Taverna On-line. For this purpose, the VPH-Share plugin has been integrated into these two platforms. Taverna Workbench is used for desktop composition and Taverna On-line is used for web composition. If the executing workflow requires user interaction, the user is able to perform such interaction via NoMachine, a web-based remote desktop application, which connects to the executing workflow.

On the server-side, the Cloud Façade is the platform where the core of the biomedical workflow is executed. This execution is controlled by exchanging information with the VPH-Share plugin integrated into Taverna Workbench or Taverna On-line. If a biomedical workflow requires interaction, a server-side component inside the Cloud Façade is started to make the remote desktop communication possible.

Additionally, a biomedical workflow can be executed directly from the Master Interface using the Workflow Manager. In this case the Cloud Façade will start its own Taverna Server, with an integrated VPH-Share plugin, and then submit the biomedical workflow to the server for execution.

Finally, an external application on the client-side can also start the execution of an already composed workflow. In such a case, the application communicates with the Cloud Façade through the Workflow Manager API. The Cloud Façade will then start its own Taverna Server, with an integrated VPH-Share plugin, and then submit the biomedical workflow to the server for execution.

In all cases, the VPH-Share plugin provides support for executing biomedical workflows with interactive and non-interactive services, as well as the execution of workflows in batch mode. The results of the execution are accessible though the LOBCDER repository, see Figure 2.

With the development of all the aforementioned processes and tools, the VPH-Share project provides the Clinical Researcher with a very versatile platform for execution and composition of biomedical workflows.

3 GIMIAS WebServices plugin to make CLP tools available as Web Services

The main goal of the GIMIAS WebServices plugin is to offer a mechanism to expose external tools (command line tools) as web services (SOAP), facilitating the integration of these tools on research workflows, providing also an interface to support tools with long execution times.

The GIMIAS WebServices Plugin, part of GIMIAS's extensions, is able to expose as a web service any processor of any GIMIAS Plugin or Command Line Plugin (CLP). A list of available Plugins and CLPs is available at http://sourceforge.net/apps/mediawiki/gimias/index.php?title=Users, and instructions on how to create new CLPs can be found at http://sourceforge.net/apps/mediawiki/gimias/index.php?title=HowToAddCommandLinePlugin.

The user just needs to activate in GIMIAS those plugins that are to be exposed and then activate the WebServices Plugin. Then, the WSDL generated by the WebServices Plugin can be used to reach the exposed Web Services.

In the VPH-Share project GIMIAS is used as a server that acts as a Web Services provider. Those services are used to compose biomedical workflows in different fields (@neurIST, euHeart, VPH-Op and Virolab flagship workflows). The composition and execution of these workflows can be done using Taverna Workbench or Taverna On-line.

Also, any CLP already exposed through GIMAS can take benefit of the improved WSDL interface to support execution with long times, in combination or not with the VPH-Share Taverna Plugin.

The list of Web Services related to the VPH-Share flagship workflows currently deployed includes:

- @neurIST services not requiring user interaction:

    - GAR segmentation
    - Geometric Descriptors computation
- @neurIST services requiring user interaction:
    - Bounding Box selection
    - Mesh editing
    - Ring cut
    - Neck selection

Other Web Services that are available (not directly related to any of the VPH-Share flagship workflows) include:

- Basic Data Visualization Capabilities
- euHeart services:
    - Cardiac Initialization
    - Cardiac Fitting
- Clinical Report Creation
- Other basic segmentation tools
    - Otsu Segmentation
    - Thresholding Segmentation
    - Region growing Segmentation

Currently GIMIAS provides a great interface to easily deploy any new external service on the VPH-Share infostructure. Interactive and non-interactive services can be provided using the VPH-Share Taverna Plugin, both locally and on-line, via the two aforementioned Workflow Management Systems.

4 Specification of services requiring user interaction

If an Appliance is to expose a service that requires user interaction, this needs to be advertised for the VPH-Share Taverna Plugin to be able to provide the support for such interaction. This can be easily configured by the Appliance developer through the Master Interface when adding the Web Service Endpoint of the Appliance in *Development Mode* (see Section 3.2.1 on document "D6.5: Production Deployment of User Access Systems") .

The configuration consists on adding a string containing the list of interactive services to be exposed by the endpoint as part of the description field of the endpoint. The string must start with the "INTERACTIVE_SERVICES=" (without quotes) and then the list of interactive services names must follow, with each service name separated from the other by a coma. For instance, one of the appliances employed in the @neurIST workflow specifies its interactive services as "INTERACTIVE_SERVICES=MeshEditing, NeckSelection, RingCut, BoundingBox" (without quotes), corresponding to the interactive services mentioned in the previous section.

Note that this process is only performed by the developer user that creates the Appliance in the VPH-Share portal (https://portal.vph-share.eu), and it is only performed once, before the Appliance is saved. Any regular user of the Appliance does not have to perform any configuration task.

The INTERACTIVE_SERVICES string is automatically read by the VPH-Share plugin during workflow execution, and when an interactive service is to be executed, the plugin detects it and provides remote desktop access to it, as explained in the next section.

5 VPH-SHARE PLUGIN

The main goal of the VPH-Share plugin is to facilitate integration of Web Services deployed on VPH-Share on scientific workflows, supporting for both composition and execution, enabling VPH-Share Web Services to be instantiated and released on demand.

In the VPH-Share project, a GIMIAS server is installed in a Virtual Machine (VM) that can be instantiated as often as needed and shutdown on demand. Each VM is called an Application and it is managed by the Cloud Façade. Many different Applications can be created exposing different sorts of services. In order to make the services available to several users at a time, several Appliances can be used at the same time.

A user can access the resources provided by the Cloud Façade by using the VPH-Share Plugin to create and/or execute a biomedical workflow. The VPH-Share Plugin integrates with Taverna Workbench for desktop workflow composition and execution; and with Taverna On-line for web-based workflow composition and execution.

When a user wants to use a service provided by the Cloud Façade in a workflow, a set of service definitions in WSDL format must be imported into the Workflow Management System being used. While normally the entries of the WSDL would contain references to a running Web Service server (endpoint), in the VPH-Share project the server is not yet running, for it corresponds to a VM. Instead, the WSDL contains an identifier that indicates the VPH-Share Plugin, which VM to instantiate and which service to execute on that machine. The URL for the WSDL of a given Appliance can be obtained from the Master Interface, using one of the *resource access* buttons (see Section 2.3.3.4 on document "D6.5: Production Deployment of User Access Systems").

When the user executes a biomedical workflow using the VPH-Share Plugin, all the complexity of the execution process is handled by the plugin in the background to make things easy for the user. The VPH-Share Plugin instantiates the needed Applications, waits for them to start up, redirects the Web Service calls to the correct Application, waits for each Application to finish its job and shuts it down when no longer needed, manages the authentication of the user in the Cloud Façade and handles possible errors during the whole process.

5.1 Desktop composition and execution tool

The architecture for desktop workflow composition and execution is presented in Figure 2. Currently, Taverna Workbench 2.4 is the main tool supported for this purpose; it is open-source and is licensed under GPL license version 2.1. This tool can be obtained from http://www.taverna.org.uk/download/workbench/2-4/.

The first step for a user to build a workflow using Taverna Workbench is to download and install the software. If the user wants to include services provided by the VPH-Share project, then the VPH-Share Taverna Plugin must be installed. For installing the plugin see Section "Installing Taverna Plugin" at http://vph-share.eu/content/vph-share-taverna-plugin.

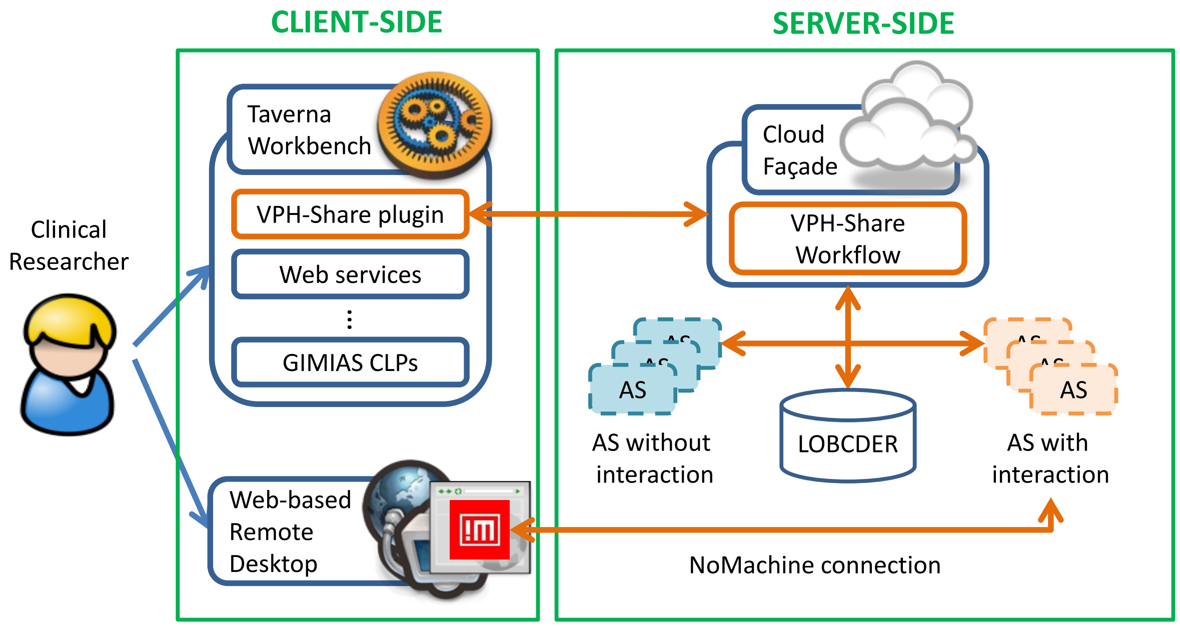


Figure 2. Desktop Workflow Management Architecture overview.

After this, the user can search the Master Interface for one or more Appliances that provide the required services. For each Appliance, the user can obtain the URL address of its WSDL, using one of the *resource access* buttons (see Section 2.3.3.4 on document "D6.5: Production Deployment of User Access Systems") in the Master Interface. This WSDL address can then be used to import the services provided by the Appliance in Taverna Workbench, see Section "Importing VPH-Share services" at http://vph-share.eu/content/vph-share-taverna-plugin.

As depicted in Figure 2, note that Taverna Workbench also supports other types of services, such as any generic Web Service, or the CLPs of a local GIMIAS installation. All these services can be combined by the user into one or more biomedical workflows. For a detailed explanation on how to build workflows with Taverna Workbench 2.4 see the user manual at http://dev.mygrid.org.uk/wiki/display/taverna/User+Manual.

When the user executes the workflow composed in Taverna, the VPH-Share plugin communicates with the Cloud Façade and creates a VPH-Share workflow, which includes all the Applications needed to execute the Taverna workflow, see Figure 2. The execution of each service in the Taverna workflow is delayed until the plugin has made sure that the Application needed for this service is launched successfully. If two or more services from the same Application are being used, only one Application is created, and then the services share the Application, saving computation resources. The services are executed in the order specified by the Taverna workflow. The output of each service, as well as the final output of the Taverna workflow, is stored in the LOBCDER, see Figure 2.

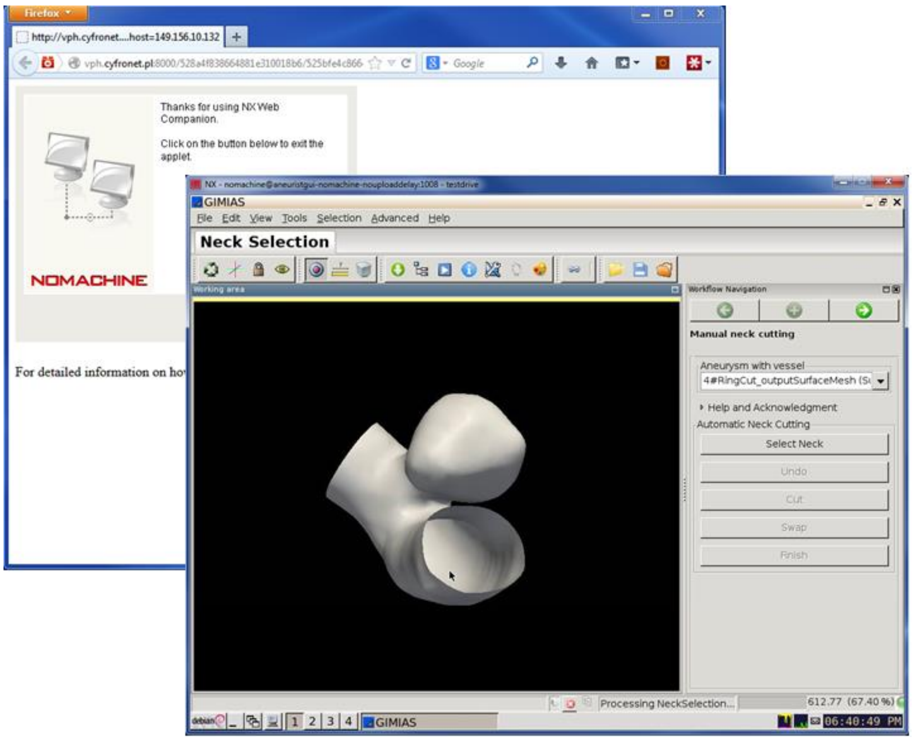

Thanks for using NX Web Companion.
Click on the button below to exit the applet.
NOMACHINE
For detailed information on ho
GIMIAS
Neck Selection
Workflow Navigation
Manual neck cutting
Aneurysm with vessel
4#RingCut_outputSurfaceMesh
Help and Acknowledgment
Automatic Neck Cutting
Select Neck
Undo
Cut
Swap
Finish
Processing NeckSelection...

Figure 3. Web-based remote desktop connection via NX NoMachine.

If an Application requires user interaction, a web browser window will automatically open in the user's desktop when the service is executed, so that the user can perform the interaction. The browser will start a web-based remote desktop session via a NX NoMachine (https://www.nomachine.com/) client. The client is available for all major platforms. The NoMachine Web Companion java applet downloads and executes the appropriate version. The end user experience is such that the remote application is run locally with a little slower response times (dependent on the network throughput) than a normal PC, see Figure 3.

An explanation on how to run an example workflow can be found at http://vph-share.eu/content/running-aneuristworkflow-short-workflow.

5.2 Web composition and execution through Taverna Online

The architecture for web-based workflow composition and execution is presented in Figure 4. Currently, Taverna On-line is supported via the High Performance Computing Online (OnlineHPC) site. OnlineHPC's online scientific workflow editor is free-of-charge and it is available at http://onlinehpc.com.

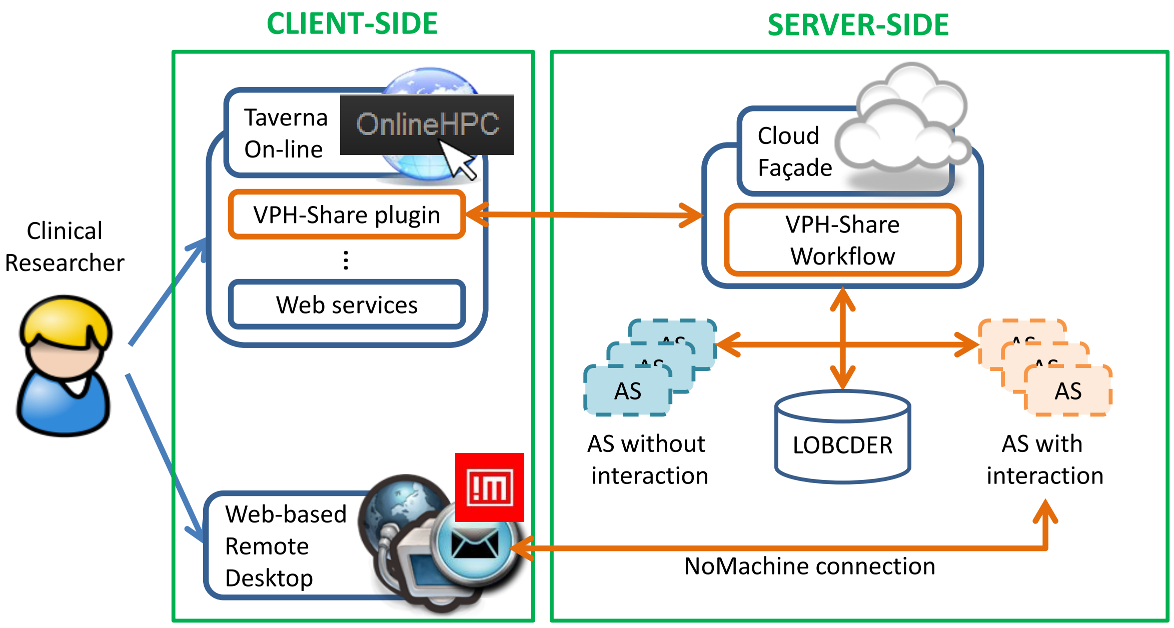


Figure 4. Web-based Workflow Management Architecture overview.

Once logged in OnlineHPC, the user just needs to press the "New Workflow" button and name the new workflow. Then, the user will be presented with a working area as shown in Figure 5. If the user wants to include services provided by the VPH-Share project, it is only necessary to click on the "VPHService" icon inside the "Processors" box in the left side of the working area, and then drag-and-drop it in the working area. The result of this process is shown in Figure 5.

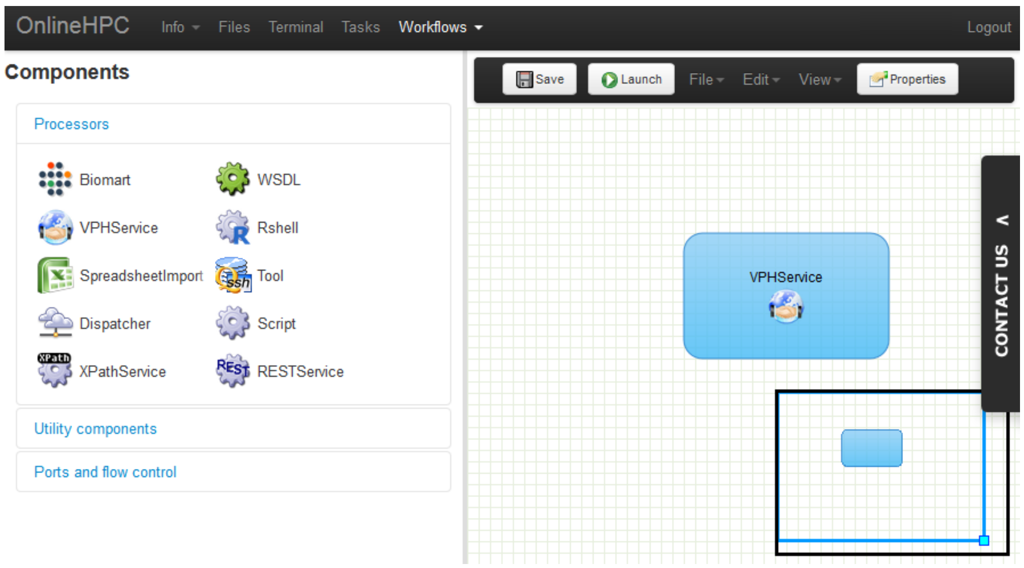
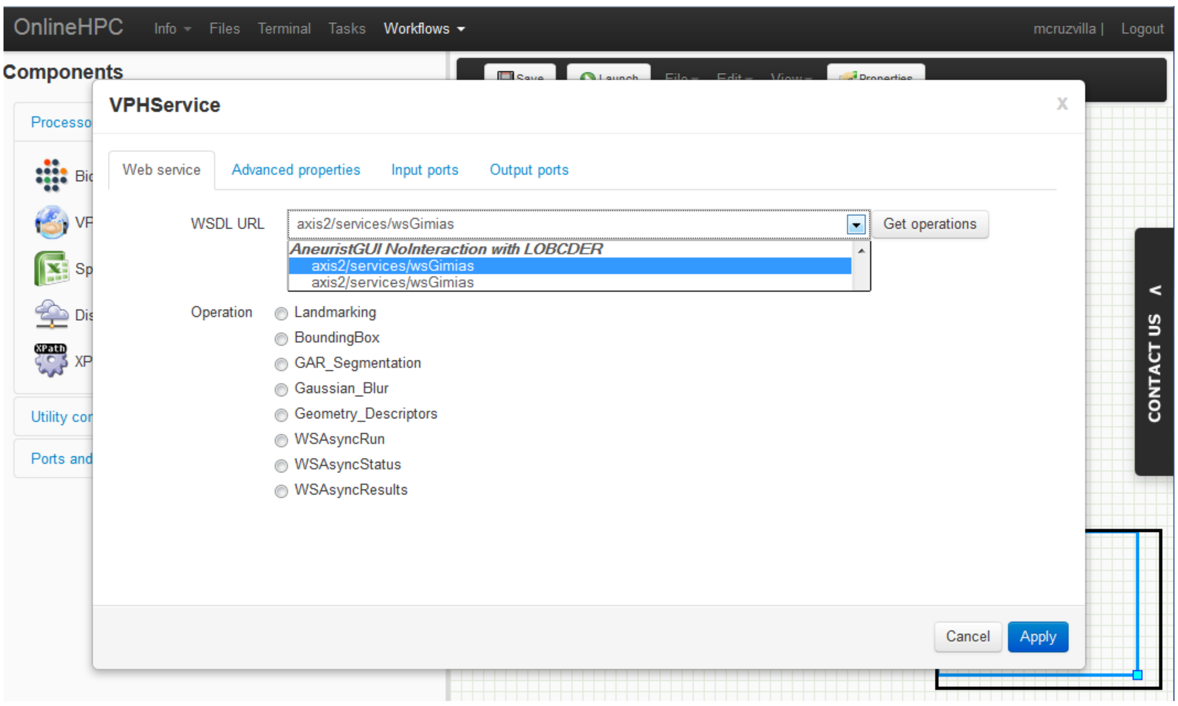


Figure 5. Taverna On-line working area.

Following this, the user must double-click on the VPHService blue box on the working area, and a configuration window will be presented. The first time this is done, the user must enter his Biomed Town credentials. Once this is done, the list of available Appliances will show up, and the user can choose an endpoint. Then, the user should press the "Get Operations" button to retrieve the list of services available in the selected endpoint. Once the list of operations is displayed, the user can choose the service needed to build the workflow. See Figure 6 for an illustration of this process. For a guide on how to build workflows with OnlineHPC, see the video at http://www.youtube.com/watch?v=0n3YhJjPBy8.



WSDL URL: axis2/services/wsGimias

AneuristGUI NoInteraction with LOBCDER
axis2/services/wsGimias
axis2/services/wsGimias

Operation:
- Landmarking
- BoundingBox
- GAR_Segmentation
- Gaussian_Blur
- Geometry_Descriptors
- WSAsyncRun
- WSAsyncStatus
- WSAsyncResults

Figure 6. Importing VPH-Share services in Taverna On-line

When the user executes the workflow composed in Taverna On-line, the VPH-Share plugin communicates with the Cloud Façade and creates a VPH-Share workflow, which includes all the Applications needed to execute the Taverna workflow, see Figure 4. The execution of each service in the Taverna workflow is delayed until the Application needed for this service is launched successfully. If two or more services from the same Application are being used, only one Application is created, and then the services share it, saving computation resources. The services are executed in the order specified by the Taverna workflow. The output of each service as well as the final output of the Taverna workflow is stored in the LOBCDER, see Figure 4.

If an Application requires user interaction, the Master Interface notification service is used to send the user a link to the web-based NX NoMachine client, which can be used to open a remote desktop session to the Application. Notifications reach the user by e-mail as well as through the Master Interface's GUI, see Section 3.2.4 on document "D6.5: Production Deployment of User Access Systems". In the case of the later, the user can click on the link and then a new web browser tab will open, automatically starting the NX NoMachine client. Then the user can interact with the Application, see Figure 3.

A short video showing the web composition and execution process is available at https://www.youtube.com/watch?v=t8D4mYKJJpY.

5.3 Support for workflows with long execution times

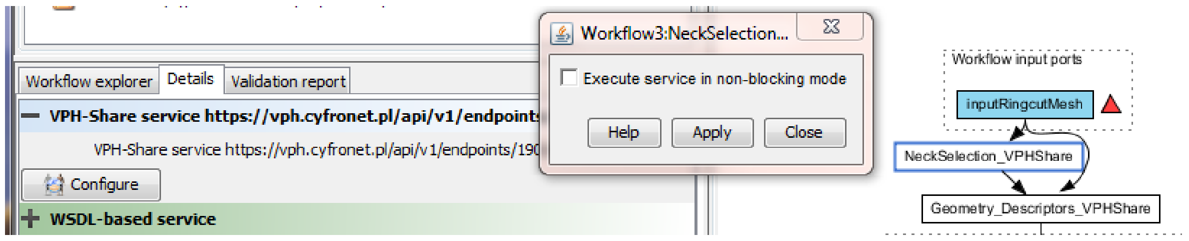
By default, the execution of VPH-Share services from Taverna Workbench or Taverna On-line has a *blocking* behaviour. This means that when the VPH-Share plugin invokes a VPH-Share service, the plugin's execution is interrupted until the service returns a response. This is perfectly valid for services that return a response promptly, but it can pose a problem for those services with long execution times. The problem is that while the VPH-Share plugin is waiting for a response, the connection to the web service is not used, as no new information is communicated from either side. Then, the connection can be taken down by any intervening web proxies or firewalls, as the communication is flagged as *timed out*. This is the typical source of errors such as "Bad gateway" or "Timeout" when running VPH-Share services.

In order to avoid this, the VPH-Share Taverna plugin has been upgraded to support a *non-blocking* communication mechanism, in which the plugin starts the execution of the service in an asynchronous fashion, and then monitors the execution status of the service every few seconds. When the plugin detects that the service is finished, it collects the response and continues the execution of the workflow. This is implemented using the following GIMIAS CLPs:

- WSAsyncRun: This method can be used to start running a CLP asynchronously. The name of the CLP and its parameters must be specified. The method returns the identification number of the process that corresponds to the running CLP.
- WSAsyncStatus: This method can be used to inquire the execution status of a CLP previously started with WSAsyncRun. The process identification number of the CLP must be specified. Possible statuses are: "STATE_PENDING", "STATE_ACTIVE" and "STATE_FINISHED".
- WSAsyncResults: This method can be used to obtain the results of a CLP previously started with WSAsyncRun, once the WSAsyncStatus method returns "STATE_FINISHED". The process identification number of the CLP must be specified. The results will be returned in the form of a string.

In this way the VPH-Share Taverna plugin makes short and simple requests to the server every few seconds, avoiding leaving the communication up for long times, and therefore avoiding any timeouts.

This new mechanism can be easily activated in a per-service base, by selecting a service and then displaying its details. In the "Details" tab, a "Configure" button will show up. By pressing this button, the configuration dialog will appear, in which the user can activate the non-blocking behaviour by clicking on the check-box next to "Execute service in non-blocking mode", see Figure 7. Moreover, this behaviour is automatically available for any CLP published using the GIMAS WebServices Plugin.



VPH-Share service https://vph.cyfronet.pl/api/v1/endpoint

VPH-Share service https://vph.cyfronet.pl/api/v1/endpoints/190

☐ Execute service in non-blocking mode

Figure 7. Configuration dialog for the NeckSelection VPH-Share service.

6 WORKFLOW EXECUTION THROUGH THE MI

Users can upload their composed Workflows to the MI, and share them with other users. Any user with access to a workflow can download it and load it into Taverna Workbench or Taverna On-line for editing and/or execution (see Sections 2.3.3.4 and 2.3.6 on document "D6.5: Production Deployment of User Access Systems"). However, if the user does not need to edit the workflow but only wishes to execute it using inputs stored in LOBCDER, then the MI provides the *Execute workflow* button, which can be used to execute the workflow in the MI. The technology behind this button corresponds to the *Workflow Manager* (WM).

Once the user enters the parameters for the execution and presses the *Initialize execution* button (see Figure 33 on document "D6.5: Production Deployment of User Access Systems"), the WM is activated behind the scenes, see Figure 8. The WM communicates with the Cloud Façade to start a new Application that runs the Taverna Server specified by the user, it waits for the server to be active and then submits to it the workflow selected by the user.

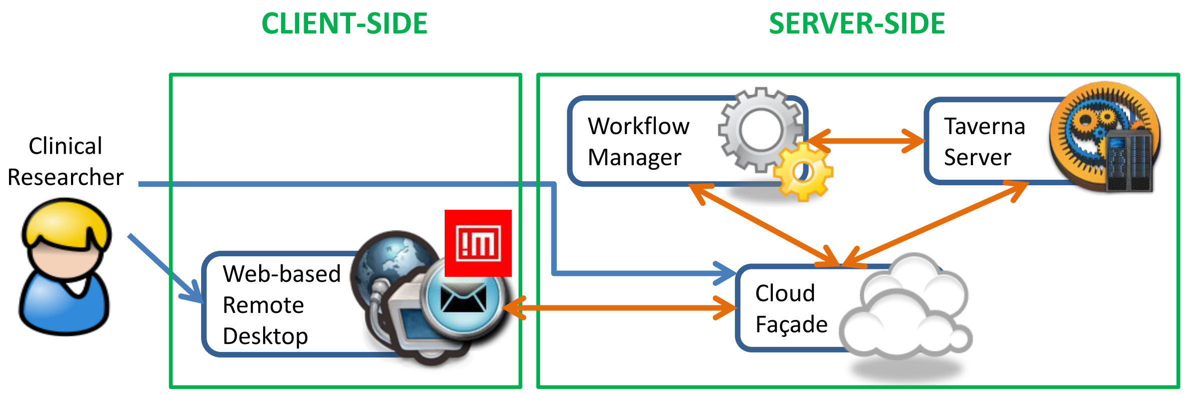


Figure 8. MI Workflow Execution Architecture overview.

If the submission of the workflow is successful, the WM proceeds to configure the workflow execution. This configuration process consist of specifying all necessary security parameters/certificates for the workflow to be allowed to run in the Cloud Façade, specifying the version and location of the VPH-Share plugin that is going to be used during workflow execution, specifying which services in the workflow require interaction and finally specifying what will be the input for the workflow, in the form of a *baclava* file. See http://dev.mygrid.org.uk/wiki/display/taverna/DataViewer+Tool for more details about *baclava* files.

In addition, when the workflow is submitted to the server, it is given a unique identification string. Before executing the workflow, the WM creates a new output folder in the LOBCDER using this identification string. Then, the WM copies into this folder all input files required by the workflow. The outputs of the workflow will also be copied in this folder. In this way, the user can easily locate the files that were produced by the execution of the workflow that he/she chose.

Once the workflow is configured, the WM indicates the Taverna Server to start running it. The WM then request the server the status of the workflow executing every 5 seconds. During this time, if the user decides to run another workflow using the same server. The WM will detect that a Taverna Server is already running and will reuse it, meaning that it will submit the new workflow to the same server, so as to save computational resources.

Once the server indicates that a workflow has finished its execution, the WM will delete that workflow from the server, releasing all resources allocated by it. By this time, the outputs of the workflow will be already copied in the output folder. If the server is not running any other workflow, then the WM will also shut down the appliance that is running the Taverna Server, then again saving computational resources.

If an Application within a workflow requires user interaction, the Master Interface notification service is used to send the user a link to the web-based NX NoMachine client, which can be used to open a remote desktop session to the Appliance. Notifications reach the user by e-mail as well as through the Master Interface's GUI (see Section 3.2.4 on document "D6.5: Production Deployment of User Access Systems"). In the case of the later, the user can click on the link and then a new web browser tab will open, automatically starting the NX NoMachine client. Then the user can interact with the Application, see Figure 3.

7 BATCH EXECUTION

With the tools created in the VPH-Share project, the user can also execute the same workflow multiple times without manual intervention. This could be used for batch execution of the same workflow with a series of input data, which is very useful for running tests on multiple subjects, performing the same experiment multiple times and other common situations in the career of clinical researchers.

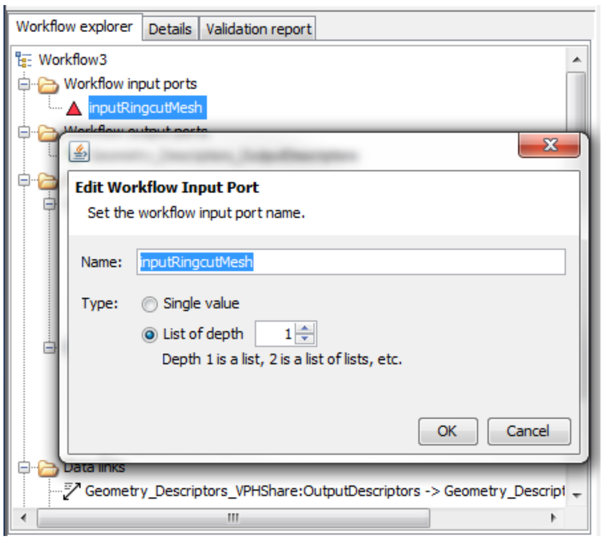
All Workflow Management Systems developed by Taverna have a built-in support for dealing with lists of data values. This means that, automatically, VPH-Share services can be input lists of values instead of single values. This later translates into what Taverna calls *implicit iterations*, see http://dev.mygrid.org.uk/wiki/display/taverna/Implicit+iteration. This is due to the fact that normally VPH-Share services have inputs of depth 0 (single values), and if the user feeds a workflow with an input of depth 1 (a list), Taverna will automatically apply the *implicit iterations* approach.

Normally, if the Taverna workflow has a single input port, this means that Taverna will perform as many *iterations* of the workflow as the number of values in the input list. That is, Taverna will execute the workflow as many times as the number of items in the input list. Each iteration takes as input one value from the input list. However, for a more detailed explanation on how "implicit iterations" are performed, especially for workflows with multiple input ports, see http://dev.mygrid.org.uk/wiki/display/taverna/Implicit+iteration.

Batch execution of workflows can be performed either in the researcher's PC or on-line, as follows.

7.1 Desktop batch execution

For activating batch execution of a VPH-Share workflow in Taverna Workbench, the user simply has to edit the input port of the workflow. In the "Workflow explorer", select the input port and press the right mouse button to open the pop-up menu. In that menu select "Edit workflow input port". In the dialog box that pops up select "List of depth" and enter a depth of 1, see Figure 9 for an example. Press "OK" to finish editing.

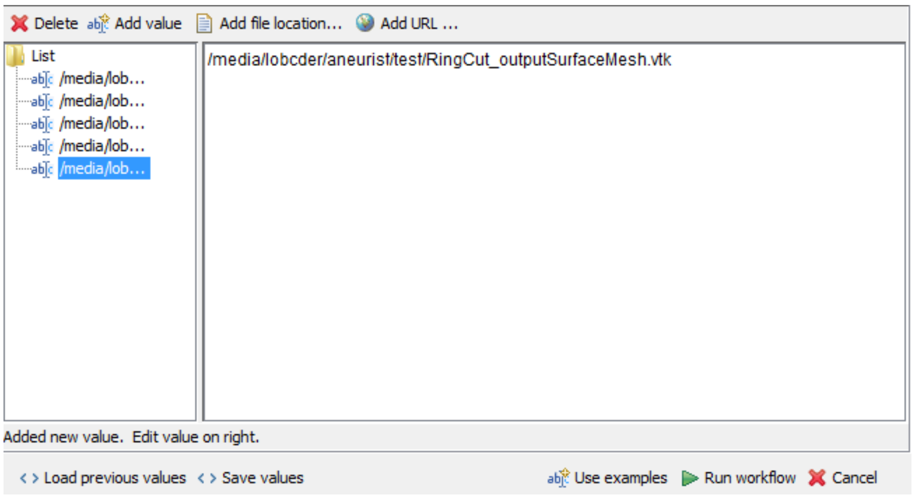


Edit Workflow Input Port
Set the workflow input port name.
Name: inputRingcutMesh
Type: Single value [✓ List of depth 1]
Depth 1 is a list, 2 is a list of lists, etc.

Figure 9. Taverna Workbench's edit input port dialog.

After this, before the user starts running the Taverna workflow, a list of values will be required as input in the Run Workflow dialog, see Figure 10. The user can enter the list manually or the "Load previous values" button can be used to load a *baclava* file specifying the list of values. See http://dev.mygrid.org.uk/wiki/display/taverna/DataViewer+Tool for more details about *baclava* files.

After clicking on "Run Workflow", Taverna will start running the workflow using the *implicit iterations* approach. During execution, in the "Graph" tab, Taverna will show progress bars and iterations numbers on each service that is performing *implicit iteration*. Similar information is shown in the "Progress Report" tab. For more details, see the "Pipelining" section at http://dev.mygrid.org.uk/wiki/display/taverna/Implicit+iteration. In addition, notice that Taverna will automatically perform *implicit parallelisation* to help increase data throughput, see http://taverna.knowledgeblog.org/2010/12/13/parallel-service-invocations/ for more details.



/media/lobcder/aneurist/test/RingCut_outputSurfaceMesh.vtk

Added new value. Edit value on right.

Figure 10. Taverna Workbench's Run Workflow dialog with input list.

For the first iteration, all the necessary Applications will be created by the VPH-Share Taverna Plugin. All following iterations will re-use the same Applications therefore saving the time and resources of continuously shutting down and re-starting the same set of Applications.

If an Application requires user interaction, a web browser window will automatically open in the user's desktop when the service is executed, so that the user can perform the interaction. The browser will start a web-based remote desktop session via a NX NoMachine client, see Figure 3. However, since in batch execution mode services are executed several times, the web-client will open only once per Application. The user must be careful not to close the browser tab, although in such a case it could be easily recovered using the browser's history. It is important to emphasize that the user must handle all the activations of the interactive service throughout all iterations, for the workflow to finish successfully.

The outputs of each iteration (intermediate and final workflow outputs) are stored in the LOBCDER, see Figure 2. However, it is important to notice that Taverna Workbench won't do any automatic renaming of the output files on each iteration. Therefore, the user must take care to input files on different locations of the LOBCDER, as otherwise the output files will be overwritten on each iteration. If the user does not want to worry about this, then the web execution tool can be used, for it will automatically create separate output folders in the LOBCDER. This is explained in the following section.

7.2 Web batch execution

Web batch execution can also be accomplished, by using the Workflow Manager (WM) available through the MI. From the point of view of the user interface, the process is exactly the same as executing a normal workflow using the *Execute workflow* button (see Section 2.3.6.2 on document "D6.5: Production Deployment of User Access Systems"). However, two preconditions are necessary to accomplish web batch execution.

The first precondition is that the workflow chosen by the user through the MI must be already prepared to accept lists of depth 1 for input, as explained in the previous section. The second precondition is that the input *baclava* file must specify a list of input values, not just one single value. The user can upload to the MI such input *baclava* and workflow definition files and run the workflow directly in the VPH-Share portal.

In addition, if the workflow is expecting a list as input, and a *baclava* file with only one input value is used, this input will be automatically converted into a list with only one item. This mean that input *baclava* files for single execution can still be used with batch execution workflows.

Another difference between single and batch web execution relates to the output folder. In single execution the WM will create one output folder in the LOBCDER, using the unique identification string of the workflow, and copy the output files directly into that folder. However, in batch execution multiple output files with the same name will be produced and this could pose a problem because the output files would be overwritten at the end of each iteration of the workflow. In order to avoid this, the WM creates subfolders within the workflow output folder. Each subfolder is named after an iteration number, and so the output files produced by each iteration will be saved inside the subfolder that corresponds to the iteration that produced them.

8 Workflow Manager API

There is also available a XML-RPC API for the Workflow Manager, which can be used to start, monitor and stop workflows in the MI using a python script. The most relevant methods available are:

- execute_workflow: This method is able to start the execution of a workflow in a particular Taverna Server instance. It takes as input the workflow definition file, the input definition file, the user credentials, the workflow title and the details of the Taverna Server to which the workflow is going to be submitted. It produces as output, among other things, the identification of the submitted workflow.
- stopWorkflow: This method stops a specific workflow, previously started with execute_workflow. It takes as input the user credentials and the identification of the workflow to be stopped.
- getWorkflowInformation: This method can be used to monitor the execution of a workflow previously started with execute_workflow. It returns information such as the

status of the workflow execution, the starting execution time, the creation time, any errors or warning messages triggered by the Taverna Server, etc. This information is also kept inside the MI for monitoring purposes.

- deleteExecution: This method stops a workflow and clears the information about the workflow stored in the MI.

With these methods it is possible to create a simple python script to execute workflows. An example pseudo-algorithm for such a script would be:

```
import wfmng

wfDefinition = open('SampleWorkflow.t2flow', 'r').read()
inputDefinition = open('SampleWorkflowInputs.xml', 'r').read()
ret = execute_workflow(userCredentials, wfTitle,  tavernaServer, wfDefinition, inputDefinition)

info = getWorkflowInformation(ret['wfId'] , userCredentials)
while info and info['status'] != 'Finished' and info['error'] != True:
  time.sleep(5)
  info = getWorkflowInformation(ret['wfId'] , userCredentials)

deleteExecution(ret['wfId'] , userCredentials)
```

9 Data Provenance and Semantic

Currently, the Taverna VPH-Share plugin also incorporates functionality for creating metadata and provenance information for every single file that is produced in LOBCDER by the plugin. That is, for every file produced in LOBCDER an entry is automatically created in the Metadata Catalogue developed by WP4, which can then be used to publish the file and perform any intelligent data search that includes the file, see WP4 deliverables for more detail. An example entry for file "Geometry Descriptors_OutputDescriptors.xml" stored in LOBCDER is shown below. If the file already exists, the metadata information is just updated.

```
<resource_metadata>
  <file>
    <author>test</author>
    <category>GenericMetadata</category>
    <creationDate>2014-02-24 15:11:42.288</creationDate>
    <description>Taverna workflow output</description>
    <globalID>df7197ce-6ae1-442b-bbb9-7a7f2e0ac530</globalID>
    <linkedTo/>
    <localID>7962</localID>
    <metadataCreationDate>2014-02-24 15:11:42.288</metadataCreationDate>
    <metadataUpdateDate>2014-02-24 15:11:42.287</metadataUpdateDate>
    <name>Geometry Descriptors_OutputDescriptors</name>
    <provenance>
      <prov:document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:prov="http://www.w3.org/ns/prov#"
            xmlns:share="http://www.vph-share.eu/ns/share#">
```

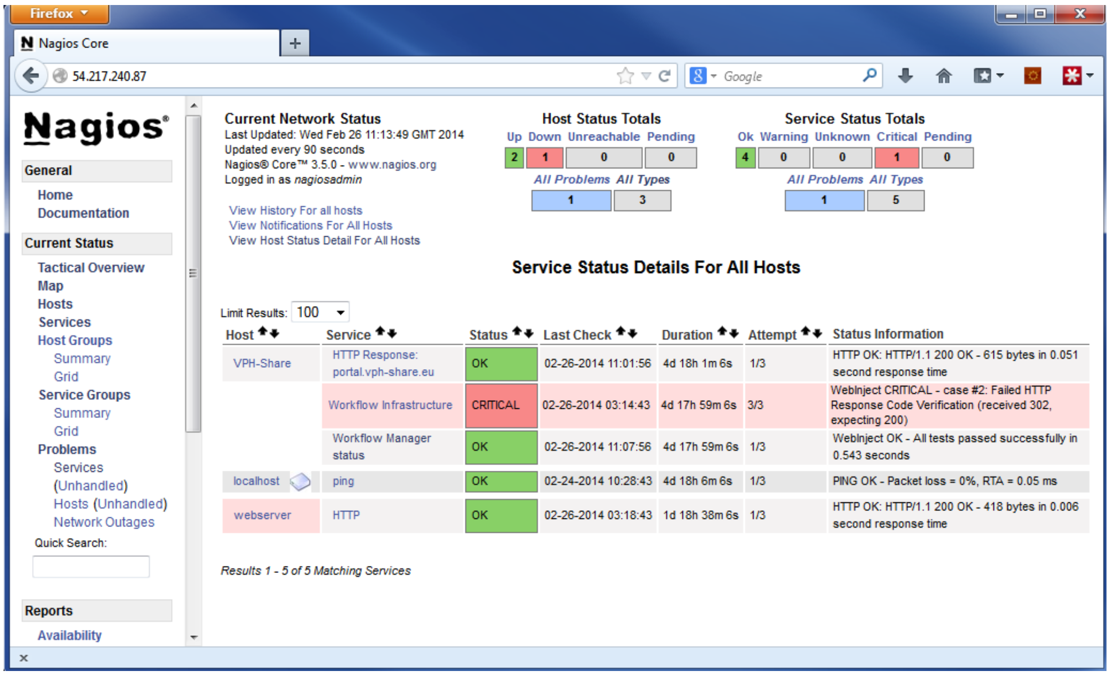
```
        <!-- Person -->
        <prov:person prov:id="ecoto">
           <prov:role>executor</prov:role>
        </prov:person>
        <!-- Webservice -->
        <prov:entity id="e3c17fe0-1657-4911-97d9-5148d78d5b74">
           <prov:label>Geometry_Descriptors</prov:label>
           <prov:location> https://vph.cyfronet.pl/api/v1/endpoints/188/descriptor
           </prov:location>
        </prov:entity>
        <!-- Input Data -->
        <prov:entity id="Dbee5df8-8c4c-46e0-a3c6-11597478d81b">
           <prov:label>inputFileNameDome</prov:label>
           <prov:location> lobcder:/aneurist/NeckSelection_outputSurfaceMesh.vtk
           </prov:location>
        </prov:entity>
        <!-- Input Data -->
        <prov:entity id="5b5a9dd2-be74-40a7-82d8-42cbc9368754">
           <prov:label>inputFileNameAneurysm</prov:label>
           <prov:location> lobcder:/aneurist/RingCut_outputSurfaceMesh.vtk
           </prov:location>
        </prov:entity>
        <!--  Entities used in generation -->
        <prov:wasGeneratedBy>
          <prov:entity prov:ref="ecoto"/>
          <prov:entity prov:ref="e3c17fe0-1657-4911-97d9-5148d78d5b74 "/>
          <prov:time>2014-02-24 15:49:57</prov:time>
        </prov:wasGeneratedBy>
        <!-- Output Data -->
        <prov:entity id="df7197ce-6ae1-442b-bbb9-7a7f2e0ac530">
          <prov:label>OutputDescriptors</prov:label>
          <prov:location>
            lobcder:/aneurist/Geometry Descriptors_OutputDescriptors.xml
          </prov:location>
        </prov:entity>
        <!--  Entities used in derivation -->
        <prov:wasDerivedFrom>
          <prov:generatedEntity prov:ref="df7197ce-6ae1-442b-bbb9-7a7f2e0ac530"/>
          <prov:usedEntity prov:ref="Dbee5df8-8c4c-46e0-a3c6-11597478d81b"/>
          <prov:usedEntity prov:ref="5b5a9dd2-be74-40a7-82d8-42cbc9368754"/>
          <prov:time>2014-02-24 15:49:57</prov:time>
        </prov:wasDerivedFrom>
      </prov:document>
    </provenance>
    <rating>0</rating>
    <relatedResources/>
    <semanticAnnotations/>
    <status>active</status>
    <type>File</type>
    <updateDate>2014-02-24 15:11:42.288</updateDate>
    <views>0</views>
    <fileType>XML</fileType>
    <format>XML</format>
    <size>429</size>
    <subjectID/>
  </file>
</resource_metadata>
```

Notice the highlighted field <provenance>, which contains a provenance document following the PROV-XML schema, see http://www.w3.org/TR/prov-xml/. The generated provenance document specifies the location of the file, its owner, its ID in the Catalogue, and all the entities that were involved in the production of the file, such as the web service that produced it and the files that were used as input to the web service. With this information, the user could re-produce the file, if needed. In the near future, the user will be able to visualize this metadata and provenance information in the Master Interface.

10 Workflow Monitoring

The first attempts towards creating a Workflow Monitoring service have been carried out, by deploying a Nagios Core engine for the project. This is a free and open source solution monitoring system, see http://www.nagios.com/products/nagioscore. The system supports the installation of specific purpose plugins, from which the WebInject plugin (http://www.webinject.org/plugin.html) has been chosen for Workflow Monitoring. The plugin will allow us to write a script with a series of steps for testing the workflow execution through the MI. Each step in the script will use the REST interface of the Workflow Manager (WM), which will be available in the near future, to execute a method that will test the functionality and availability of the WM.

At the moment, three test services have been setup. See the three services under hosts "VPH-Share" in Figure 11. The first one just checks that the portal.vph-share.eu returns a valid HTTP response, which indicates that this web page is up and reachable. The second and third services use the WebInject plugin, so each one runs a test script using operations. The script of the second service has been set to fail on purpose, so as to test the reaction of the server to this failure. Notice how the failed service is highlighted and its status is set to "CRITICAL". The system administrator received an alert e-mail when the service failed. The script of the third service has been set to be successful. Notice how the service's status is set to "OK".



Current Network Status
Last Updated: Wed Feb 26 11:13:49 GMT 2014
Updated every 90 seconds
Nagios® Core™ 3.5.0 - www.nagios.org
Logged in as *nagiosadmin*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
2	1	0	0

All Problems	*All Types*
1	3

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
4	0	0	1	0

All Problems	*All Types*
1	5

Service Status Details For All Hosts

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
VPH-Share	HTTP Response: portal.vph-share.eu	OK	02-26-2014 11:01:56	4d 18h 1m 6s	1/3	HTTP OK: HTTP/1.1 200 OK - 615 bytes in 0.051 second response time
	Workflow Infrastructure	CRITICAL	02-26-2014 03:14:43	4d 17h 59m 6s	3/3	WebInject CRITICAL - case #2: Failed HTTP Response Code Verification (received 302, expecting 200)
	Workflow Manager status	OK	02-26-2014 11:07:56	4d 17h 59m 6s	1/3	WebInject OK - All tests passed successfully in 0.543 seconds
localhost	ping	OK	02-24-2014 10:28:43	4d 18h 6m 6s	1/3	PING OK - Packet loss = 0%, RTA = 0.05 ms
webserver	HTTP	OK	02-26-2014 03:18:43	1d 18h 38m 6s	1/3	HTTP OK: HTTP/1.1 200 OK - 418 bytes in 0.006 second response time

Results 1 - 5 of 5 Matching Services

Figure 11. Nagios Core web interface showing VPH-Share service monitoring