Grant agreement no. 283562

N4U - neuGRID for you
Expansion of neuGRID services and outreach to new user communities

Combination of Collaborative Project and Coordination and Support Action
Objective INFRA-2011-1.2.1 – e-Science environments
Start date: July 1st 2011 **- Duration**: 42 months

Deliverable Data:
Deliverable reference number and title: D10.2 - Interim Report on the Delivery of the User Analysis Environment
Due date: Month 24, 30 June 2013
Actual submission date: 10/07/2013
Organisation name of lead contractor for this deliverable: P3 UWE
Dissemination level: Public

Authors:
Mr Andrew Branson (P3 UWE)
Dr Jetendr Shamdasani (P3 UWE)
Prof Richard McClatchey (P3 UWE)

Approval
Work Package Leader: Andrew Branson (P3 UWE)
Area Leader: S. Liaquat
Project Coordinator: G.B. Frisoni (CO1 FBF)
PMT members: G.B. Frisoni (CO1 FBF), D. Manset (P2 maatG), S. Liaquat (P3 UWE)

Version: 1

Document History

Issue	Date	Comments
First draft	28/06/2013	Document circulated internally to all partners for comments
Second draft	08/07/2013	Edits and comments received by R. van Schijndel (P12 VU/VUmc) and C. Bagnoli (CO1 FBF)
Third draft	08/07/2013	Comments and edits addressed by A. Branson
Foruth draft	09/07/2013	Final editing by CFc
Version 1	09/07/2013	Document finalized by A. Branson

Table of Contents

Executive Summary

The purpose of WP10 is to develop a user environment for managing, executing and recording scientific workflows run on neuroimaging datasets, known as the N4U Analysis Service (also known as the Analysis Suite or Analysis Tools). The Analysis Service provides workflow orchestration for scientists and a platform for them to execute their experiments on the GRID. As well as providing the possibility for workflow execution, it will allow users to recreate their experiments on the N4U Infrastructure using previously recorded provenance information as well as a set of visualisation tools allowing users to view their results and perform statistical analyses. Later in the WP10 lifecycle, social networking functionality will be provided to the N4U user community allowing users to share their experiments and results with others.

This is Deliverable 10.2 from the N4U project entitled "Interim Report on the Delivery of the User Analysis Environment". The purpose of this deliverable is to describe the work that has been carried out by WP10 between Months 18 and 24 of the project. The majority of the "plumbing" or backend work for the WP10 Analysis Tools is complete, a sample pipeline and dataset have been registered, and a command-line interface has been developed for users to create and execute analyses. These set of tools are considered to be in an ***Alpha*** stage and are ready for testing, feedback and feature requests from the N4U community. The tools, as described here, are available on the N4U infrastructure and are ready to be used by the users in N4U.

Intended Audience

The N4U consortium, workpackage leaders, and end users of the N4U Analysis Service.

1. Introduction

The purpose of this document is to report on the status of the development of the Analysis Suite which is the main goal of Work Package 10 of the N4U project. This document is an extension of document D10.1 (Specification of the User Analysis Environment). This prior document presented a service architecture which specified the interaction between the services which are currently being developed. During the course of development, some changes have been made to the previous architecture. These changes were necessary because the previous design was a high level overview and description of the services which are used in N4U. However, now that the initial implementation has been completed, the developers on WP10 are able to comment on the current aspects of the Analysis Service Suite in more detail. It should be noted that this document describes the current state of the WP10 suite of tools for developers of N4U to use. Therefore, these tools are currently in an ***alpha*** state and will change according to incoming requirements, for the forthcoming beta release.

In the previous deliverable (D10.1), an evaluation of different systems to base the WP10 services on was performed. From the outcome of this evaluation, the CRISTAL [1] system was chosen as the system of choice. This is was due to its *provenance* capturing capabilities, since CRISTAL was designed with provenance in mind. This means that everything in CRISTAL (workflows, activities, scripts, schema definitions) is versioned and changes are recorded. The strength of this approach is that users of CRISTAL are able to reproduce, examine, and trace versions of their analyses. A full discussion of CRISTAL and what it provides for the WP10 Analysis Suite is out of the scope of this document. The interested reader is directed to the previous deliverable (D10.1) or [1].

Section 2 describes the advancements in WP10, thus far discussing the current WP10 architecture, the CRISTAL Model and the implementation around it. Section 3 describes the user interaction and usage of the WP10 clients. Section 4 describes the future work to be conducted towards the beta release with a timeline of release points, and finally section 5 concludes this document.

2. Advancements

This section describes the advancements that have been done in WP10 in regards to implementation of the analysis tools. These include the WP10 Architecture, the CRISTAL model and the notion of Provenance Enabled Items. The first subsection begins with the WP10 Alpha Architecture.

2.1 WP10 Architecture

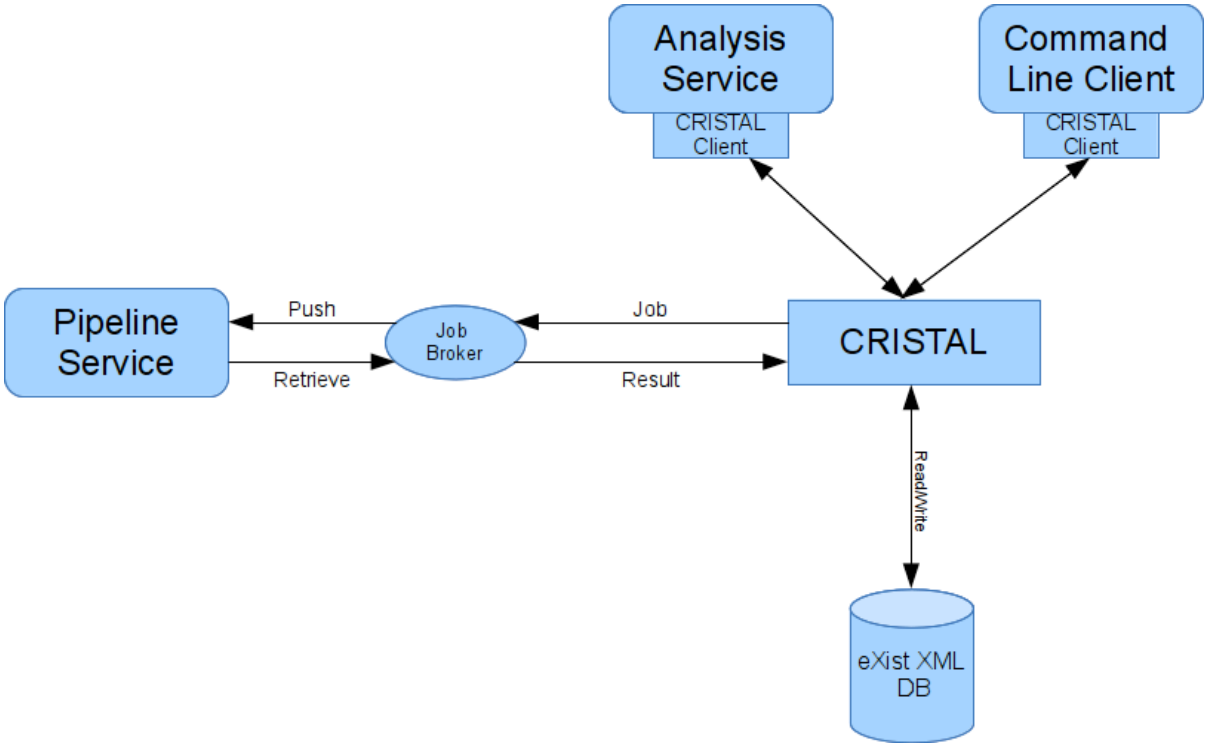

Figure 1 - The WP10 Alpha Architecture

The Analysis Service and Command Line Clients in figure 1 show interaction points for the user(s) of N4U to be able to access the CRISTAL software. These are described later in sections 3.1 and 3.2. However, as described in the previous deliverable (D10.1), the CRISTAL software was chosen as the "core" for the services deployed in WP10. This is because in comparison to other workflow based systems, CRISTAL is able to provide *provenance by design*. A new addition to the above diagram is that of a *Job Broker* component. This component is the sole point of interaction with the Pipeline Service from WP5. The purpose of the job broker is to feed the workflow orchestration calls from CRISTAL into the Pipeline Service by using the Job Broker. The format in which the Job Broker submits calls to the GRID (via the Pipeline Service) is the same format that ExpressLane (D5.2) uses to submit jobs to the WP5 Pipeline service. This is primarily in reference to the "diligence.sh" script, which handles the execution environment for ExpressLane scripts on the GRID. The decision to use

the diligence.sh script on the Computing Elements (CE) to execute analyses via the pipeline service, was to simplify the deployment of the usage of already existing algorithms. The script is a "wrapper" script which creates a tarball with the following four parameters : 1) Current Date and Time, 2) the output directory where the results are stored on the GRID, 3) the script name for example ReconAll.script and 4) finally the dataset. The populated parameters look like this :

```
1)20130605-152317
2)lfn:/grid/vo.neugrid.eu/home/DC=eu_DC=europa_OU=Organic_Units_OU=Users_CN=k
hawar_CN=6693_CN=Joe_Bloggs/expressReconOutput
3) ReconAll.script
4) 130_S_1337-SAG_MP_RAGE-2007_03_05_15_59_19.0-S27584.zip
```

As well as executing the diligence.sh script on the GRID with the above parameters, a tarball is passed onto the GRID via the pipeline service (InputSandbox.tar.bz2). Using ReconAll as an example this tarball contains the following files :

- ReconAll.env - Extra environment variables that need to be set up on the computing element (CE) for custom execution.
- common.sh - Common BASH functions that are used by the scripts on the CE.
- ReconAll.script - The script file which runs the scripts on the CE.
- ReconAll.express - The extra parameters that are set up for the ReconAll.script to run.

2.2 CRISTAL Model

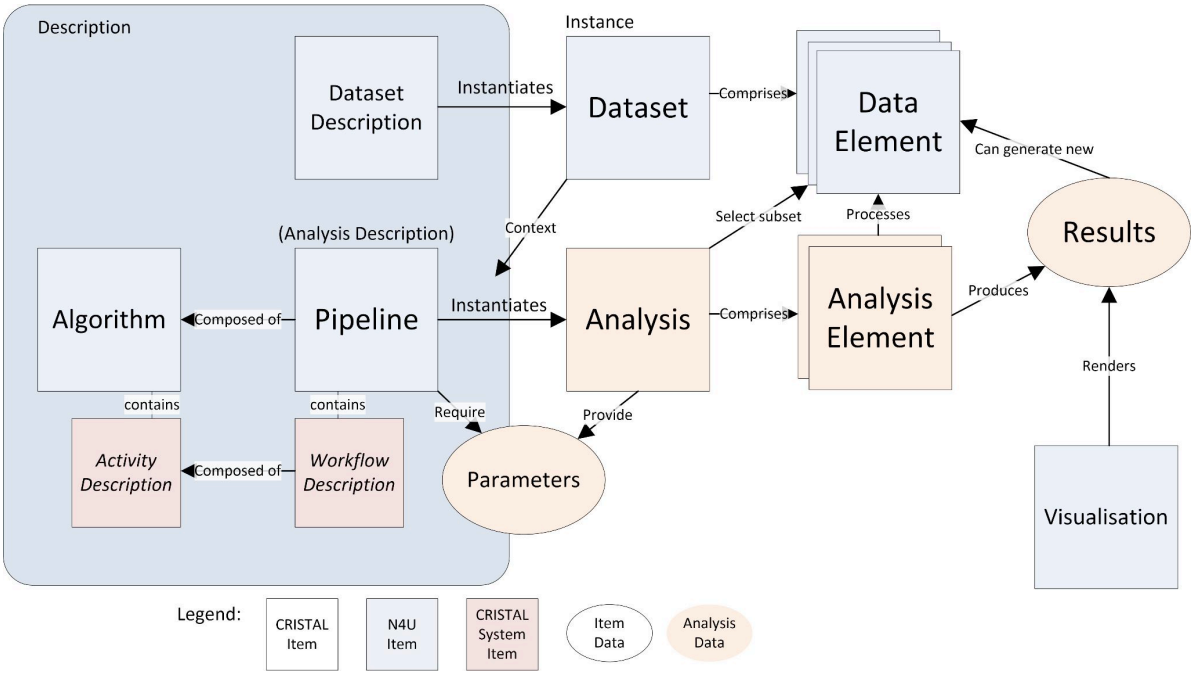

Figure 2 - Updated CRISTAL Model

As well as the addition of the Grid Broker Concept into the WP10 codebase, the CRISTAL model has also been evolved during implementation to the new model, shown in figure 2. New concepts have been added, *Data Elements* and *Analysis Elements*. This is a step towards the *Provenance Enabled Objects* described in the next section. Here the model has been split between its Description and Instances. The diagram shows five different types of *Items* (description-driven CRISTAL objects). These items are CRISTAL internal items (white), N4U Specific Items (blue), CRISTAL System Items (Red), Item Data (white ovals) and Analysis Data (light brown ovals and squares).

The model contains Items holding metadata on the pipelines and datasets registered in the N4U Information Services. **Pipeline** Items give the location of the ExpressLane script that should be run, along with default execution environment settings, and any common directory locations that should be passed with the job. **Dataset** Items contain **Data Elements**, which are sets of files that should be processed together in one job, along with specific metadata about that set that can vary in its composition between different datasets. For instance, the elements of the OASIS dataset contain a 'Scan Type' property that indicates whether that scan was taken longitudinally or as a cross-section. In the alpha release, this information is minimal, but in later releases this metadata will be expanded to include subject clinical data where available. The Data Element concept replaces the 'File' Item type of the previous version of the model, which would only have referenced one file.

The central Item type of the Analysis service itself is the '**Analysis**', which is a user-initiated process of executing a pipeline on one or more elements of a single dataset. Analyses are instantiated from Pipeline Items, with a Dataset and some Data element IDs given as parameters. In a later release, Pipeline items will be able to require additional parameters, that must or may be given when the user creates the Analysis to override or append the defaults included in the Pipeline specification. Each Analysis belongs to the user that created it, and can only be viewed by that user. Then, the Analysis suite instantiates an **Analysis Element** for each given Data element, which creates an instance of the Pipeline workflow that can be dispatched to the Grid as executable jobs.

Each job is processed by the **Grid Broker**, which packages up all necessary files and parameters into an archive that will be uploaded with the job submission to a Grid computing element. The broker then monitors those jobs, and updates the corresponding Analysis element as its execution status changes. As each pipeline instance completes, the execution console and error output are stored in the Analysis element, which is marked as complete. Any result files should have been uploaded to the user's Grid home directory by ExpressLane at the end of execution on the computing element. The parent Analysis object keeps track of the number of completed Analysis Elements, until it is also marked as complete when they are all done. Later, the **Visualisation Service** would be invoked at this point, to create a visual report of the execution as required.

2.3 Provenance Enabled Objects

CRISTAL is a system that records every change made to its objects, which are known as Items. Whenever a modification is made to a piece of data, the definition of that piece of data, or application logic, that change and the metadata associated with that change (who did it and when)

are stored alongside that data. This makes CRISTAL applications fully *traceable*, and this data may be used to assemble detailed *provenance* information. In N4U, CRISTAL holds data from the Analysis service, containing the full history of computing task execution, but it can also provide this level of traceability for any piece of data in the system, such as the datasets, pipeline definitions and queries.

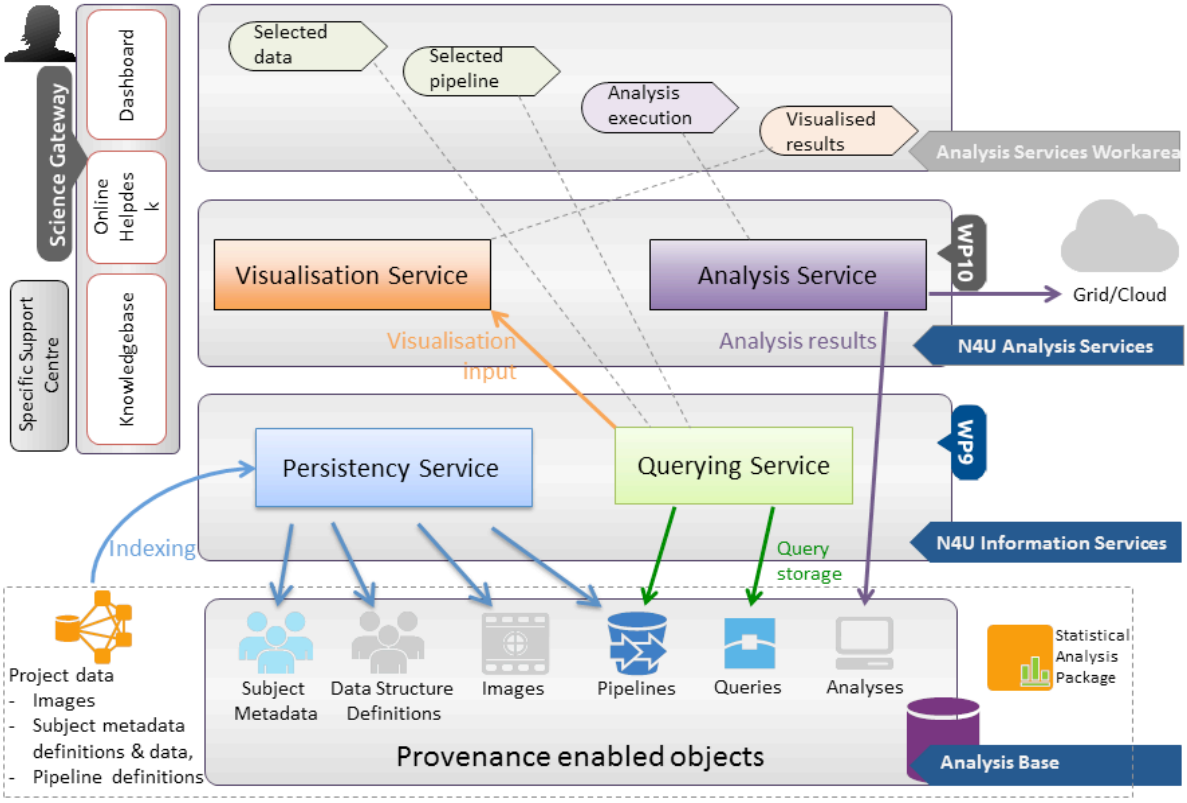

Figure 3: WP10 in N4U

Figure 3 shows where the work of WP10 (with some links to WP9) fits into the grand scheme of the N4U project. For the purpose of integration with other work packages, such as WP5 and WP9, the WP10 N4U Analysis Services "sit on top of" the WP9 services. Here, WP10 is represented by the Analysis Service (AS), the Visualization Service (VS) and the Analysis Service Work-area (ASW). Here, the key thing to note is the notion of *Provenance Enabled Objects.* These are all pieces of data that are stored within the N4U project. However, each of these pieces of data has provenance information implicitly attached. This provenance is data about the changes made to these objects and along with the changes (history) there are different versions of this data stored within N4U, therefore, nothing is deleted. These provenance enabled objects are *subject meta-data, data structure definitions, images, pipelines, queries and analyses.* In the current state of N4U, especially in the context of this ***alpha*** release, only Analysis provenance has been considered. This analysis

provenance concerns the history of a change to an Analysis Object within the Analysis Service. The provenance format used is the CRISTAL event definition which was described in the previous deliverable (D10.1) and in [1]. An example of the job execution provenance that is stored in CRISTAL is shown below:

```
<GridJobStatus>
  <workflowID>
      [GridWorkflowHandler]-
{[wms://wms.maatg.eu:7443/glite_wms_wmproxy_server]-
[https://lb.maatg.fr:9000/Pybe2vLr7KLjlhpnUG9NyQ]}
  </workflowID>
  <state>DONE</state>
  <exitCode>0</exitCode>
  <reason>null</reason>
  <startDate>2013-06-25T00:40:44.000+01:00</startDate>
  <createdDate>2013-06-25T00:37:27.000+01:00</createdDate>
  <endDate>2013-06-25T00:49:19.000+01:00</endDate>
  <stdout><![CDATA[before : /opt/exp_soft/neugrid (standard out information)
                                        ]]></stdout>
  <stderr><![CDATA[Using grid catalog type: lfc (standard error information)
]]></stderr>
  <executionHosts>[ng-maat-server9.maatg.eu:8443/cream-pbs-neugrid, ng-maat-
server8.maatg.eu]</executionHosts>
</GridJobStatus>
```

The above XML fragment is the output of the Job Broker (figure 1). This is the information that is stored in CRISTAL. Here, the "workflowID" is stored as the unique identifier for this object. The state is stored, in this case DONE. The exitCode (0 for successful completion and any other number for unsuccessful) gives the reason the job terminated if there is an error. The startDate which contains the date and time the job was started on the GRID. The createdDate which shows the date and time the job was submitted on the GRID. The endDate date and time states when the workflow was completed. The stdout (standard out) and stderr (standard error) Grid job output is stored as plain text and finally the executionHosts (which CE(s) ran the job/workflow). CRISTAL has a backend XML Database based on eXist-db [2]. This XML fragment is returned by the job broker then passed onto CRISTAL for storage. In CRISTAL terminology, this is the *event* that is recorded about the execution. This fragment is fully queryable via CRISTAL using XQuery. Therefore, once this record is stored, it can be accessed and analysed at any time. For the forthcoming *beta* release of the Analysis Suite the full list of provenance enabled objects (figure 3), shall be defined and stored in CRISTAL in a similar fashion. These would model and store the interaction with WP9's Persistency Service to store user specific datasets and the modification of the Querying Service to query the CRISTAL XML backend.

3. Interaction and Usage

3.1 Command Line Client

The current recommended means of accessing the N4U Analysis Service is the command-line client installed in the Pandora shell environment on the N4U infrastructure. This can be accessed either through an XTerm session on Desktop Fusion, or through the GateOne terminal portlet. The client application can function either as an interactive shell, or as a script batch file processor. This allows more *advanced* users of the N4U infrastructure to write custom scripts and programs to deploy their workflows within N4U.

The client is invoked from the prompt with the command:

```
> n4u-analysis
```

this starts the interactive shell.

It is invoked in batch mode with:

```
> n4u-analysis -script myscript.js
```

this will run through the given script file executing the commands, then exit. The filename and extension are not significant.

In either mode, the interpreter is a JavaScript interpreter running in a CRISTAL client environment. Other languages may be supported later if desired. On start-up, the client assumes the identity of the currently logged in GRID user, which will have a CRISTAL account created for it if it doesn't already exist. The user then has several functions and API Objects available to interact with to create, start and monitor their analyses. These are the following:

Shell commands

- **help()** - prints out the current N4U Analysis Service CLI documentation.
- **exit()** - terminates the shell

Dataset discovery

- **listDatasets()** - lists datasets available for analyses
- **listDataElements("dataset")** - list the data element IDs of the given dataset.
- **showDataElement("dataset", "id")** - prints out the XML data associated with that dataset element.

Pipeline discovery

- **listPipelines()** - list available pipelines
- **myPipeline = loadPipeline("name")** - load a pipeline object of the given name and keep it as the variable 'myPipeline' for further use .
- **showPipeline("name")** or **myPipeline.getDetails()** - show XML data of a pipeline object

Analysis creation

- **listAnalyses()** - list all of the current user's analyses by name
- **myAnalysis = createAnalysis("name" or myPipeline variable)** - create new analysis from the given pipeline and keep it in the local variable 'myAnalysis' for configuration and execution. It will not be registered in the database until it is started.
- **myAnalysis.setDataset("name")** - assign a registered dataset to this analysis. Note that this is will not assign every element of this dataset to the analysis. Elements must be assigned individually.
- **myAnalysis.addDataElement("id")** - add a data element from specified set to this analysis. An error will be reported if the element is not found in the registered dataset.
- **myAnalysis.start()** - create the analysis in the analysis database, instantiate its pipelines for each dataset elements, and start submitting its jobs to the Grid. Once the analysis has been created, its generated unique name will be printed to the console.
- **myAnalysis.getName()** - display the unique name of the running analysis, for later loading.

Analysis monitoring

- **myAnalysis = loadAnalysis("analysis")** -- load existing analysis.
- **myAnalysis.listElements()** -- list analysis elements
- **myAnalysis.listDataElements()** -- list data elements assigned to this analysis
- **myAnalysis.status() or status("analysis")** - show analysis execution progress. Each analysis element is reported separately, along with the number of completed elements, and a percentage complete for the entire analysis.
- **myAnalysis.getResults()** or **results("analysis")** - print result for all completed analysis elements
- **myAnalysis.relaunch()** - start a new analysis with the same pipeline and data as this one
- **statusAll()** - report status of all analyses owned by the current user

As can be seen the functions available in this client are complete. An example usage of the client can be found in the following code fragment (the numbers denote line numbers and are not function calls):

```
1. newAnalysis = createAnalysis("nG+FreeSurfer+5.2.0+ReconAll+v01");
2. newAnalysis.setDataset("OASIS");
3. newAnalysis.addDataElement("nG+OASIS+OAS10032MR1");
4. newAnalysis.addDataElement("nG+OASIS+OAS10037MR1");
5. newAnalysis.start();
```

in this fragment, the first line creates new Analysis Object using the createAnalysis function, the parameter passed to this is the name of the algorithm on the GRID. In this case, it is the ReconAll script. The second line defines which dataset this algorithm should be run on, for example the OASIS dataset. The third and fourth lines specify two elements for the analysis to process, these are individual pieces of data in the larger dataset. The final line instantiates the analysis in CRISTAL, which will result in its jobs being submitted to the GRID.

When the analysis has been created, its name is reported on the console:

```
Analysis nG+FreeSurfer+5.2.0+ReconAll+v01.OASIS_1372324182113 created
```

This name is generated from the pipeline name, the dataset name, and a timestamp, and may be kept for later querying of status and results. The analysis may be queried at any time:

```
N4U> newAnalysis.status()
Analysis nG+FreeSurfer+5.2.0+ReconAll+v01.OASIS_1372324182113 contains 2
elements. 0 are complete (0%)
 - Element
nG+FreeSurfer+5.2.0+ReconAll+v01.OASIS_1372324182113+nG+OASIS+OAS10032MR1:
QUEUED since 2013-06-27 10:12:08
 - Element
nG+FreeSurfer+5.2.0+ReconAll+v01.OASIS_1372324182113+nG+OASIS+OAS10037MR1:
QUEUED since 2013-06-27 10:12:11
```

And once the analysis is reported as complete, the details of the run can be retrieved:

```
N4U> newAnalysis.getResults()

Analysis nG+FreeSurfer+5.2.0+ReconAll+v01.OASIS_1372324182113 contains 2
elements. 2 are complete (100%)
nG+FreeSurfer+5.2.0+ReconAll+v01.OASIS_1372324182113+nG+OASIS+OAS10032MR1:
<GridJobStatus>
      <workflowID>[GridWorkflowHandler]-
{[wms://wms.maatg.eu:7443/glite_wms_wmproxy_server]-
[https://lb.maatg.fr:9000/MXjM5rUVnylvF0PDyX36KQ]}</workflowID>
      <state>DONE</state>
      <exitCode>0</exitCode>
…&c
```

The resulting outputs should also now be found in the user's home directory on the Grid.

3.2 Web Service Description

The N4U Analysis Suite currently only has a basic web service interface for access. The recommended usage of the Alpha functionality is the command line interface (see previous section). However, along with the command line interface there is an alpha web service based on Apache CXF [3], which is able to be used by more advanced users in their applications. This web services provides a way for users to programmatically use the analysis service in their applications. Since it has been developed as a web service it is language neutral on the part of the client. It contains the following functions :

***AnalysisID* createAnalysis(pipelineID)** - This is for a user to create a *new* analysis from an existing pipeline. The input parameter it he ID for the pipeline and the return type is an Analysis ID.

***Started* startAnalysis(int analysisID)** - This starts an already created analysis in the past. Here the input is the Analysis ID and the output is the job id of the running analysis if successfully started or an error message.

***Status* queryAnalysis(int analysisID)** - This allows the user query the current state of an analysis. Here the output would be the current state of analysis with the analysis ID as the input.

Boolean abortAnalysis(int analysisID) - This will also the a user to start/stop an already running Analysis. It will return a Boolean value stating if the abort was successful.

4. Future work and Beta Release Plan

4.1 Provenance in N4U

It is envisaged in the N4U project to have three critical areas where users can interact with provenance information. The main areas that WP10 and as a side effect, WP9 consider are *Data Provenance*, *Analysis Provenance* and *Provenance Interoperability.* These are the core areas that we consider to be essential to the N4U project. It is foreseen for the beta release that data provenance will be handled by the *Persistency Service* (WP9). The analysis provenance is currently being handled by the *Analysis Service* (WP10) and the provenance interoperability will be handled by the *Provenance Service* (WP10). Previously in this document, it was described that everything in the WP10 "core" consists of provenance enabled objects. This means that all object types in N4U which are stored in the WP9/10 data store have provenance by design (section 2.3). This makes the storage of data provenance for the future beta release a much simpler task since the provenance mechanism is the same as the current Alpha Analysis Service. However, what has not been discussed previously is the provenance interoperability aspect present within the N4U project. This effort will be available for the forthcoming beta release and is seen as a means to export the provenance data present in N4U to a standard format. Currently, we are working on exporting the provenance enabled objects to Open Provenance Model (OPM). This will allow N4U users to use their provenance data in other OPM compliant systems.

4.2 Beta Release Functionality

The planned functionality for the beta release is the following:

Graphical User Interface for the Analysis Service

This will be a graphical frontend (GUI) for the analysis web service described previously in section 3 and will have similar functionality as the command line client. This front end will be based on the currently deployed Liferay portlet environment. This will be a fully functioning environment built

on top of the current analysis web service. It will provide an area where users will be able to submit jobs to the infrastructure using a full set of GUI tools.

Support for different ExpressLane files

Currently in the Alpha release we have only tested the alpha services with a small subset of ExpressLane (D5.2). However, in the future beta version, WP10 will support all the ExpressLane scripts.

Dataset Support

As well as support for the updates / new algorithms in ExpressLane, the Analysis service will be able to support new datasets coming into the N4U infrastructure by the Persistency service import mechanism of WP9. This will be accomplished with the aid of the WP9 team.

Provenance Service

As mentioned previously, this service will facilitate interoperability of N4U usage data by providing a provenance export scheme. As mentioned previously, this will be an OPM-based export of the N4U provenance data. This will also allow users to reuse their N4U analyses in other OPM compliant applications.

Analysis Suite Test Framework

This is an internal feature which will be provided to developers to test their scripts before deployment on the analysis suite.

Documentation

Thorough documentation will be provided on the usage of the Analysis Suite tools. This is in conjunction with the amended Description of Work where WP10 will provide an extra deliverable concerning the beta version of the Analysis Suite.

4.3 Beta Release Timeline

The N4U Beta Analysis Suite of Tools will be feature complete and be ready by Month 36 of the project, this is so that they can be used by the N4U users and either bug reports or feature requests can be filed. The release timeline is currently as follows:

Feature	Month	Description
Basic Portlet Based User Interface	M27	This will be the basic GUI for the Analysis Suite. This will be based on the Liferay portlet API that is provided by the infrastructure.
Alpha Provenance Service	M30	This will be an initial version of the Provenance Service (detailed earlier). It will be released on the infrastructure for people to comment on.
Full Express Lane support	M32	This will support all ExpressLane scripts (not just ReconAll) This functionality is there already, but, has not been tested outside of the example scripts. However, aggressive testing will be done.
Beta Analysis Suite (Service)	M36	This will be the fully functional set of analysis tools included the BETA GUI and Provenance Service. Users will be able to comment and file bug reports using the N4U bug tracker.
Beta Release Documentation	M36	This will be the initial documentation on how to use the Analysis Suite.
Final Analysis Suite	M42	This will be the final, bug fixed version of the Analysis Suite.
Final Documentation	M42	The final document that WP10 shall produce.

5. Conclusion

This document has presented the current state of the WP10 Analysis User Environment. As can be seen, the majority of the backend plumbing work has been completed. This has included the interaction of the WP10 Alpha Analysis Suite and the WP5 and WP9 services. Currently users can submit their ExpressLane jobs via CRISTAL to the pipeline service and the provenance / output of these jobs are stored in CRISTAL and the Analysis Base.

In the future, support will be added for the rest of the provenance enabled objects from figure 3, since this feature is crucial to the success of the project. This is the immediate concern of WP10 and support is being added at the moment. Currently the only means to access the Analysis Service in WP10 is either via the command line client or the web service. However, a basic GUI will be ready

to be demoed at the end of September. A release schedule has also been presented with deadlines, which will be adhered to by the WP10 developers.

References

[1] A. Branson et al. “Evolving Requirements: Model-Driven Design to Support Change” Journal of Information Systems, Currently Under Review.

[2] eXist XML DB - http://www.exist-db.org/

[3] Apache CXF - http://cxf.apache.org/