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: D10.1 Specification of the User Analysis Environment
Due date: Month 18, December 31, 2012
Actual submission date: December 24, 2012
Organisation name of lead contractor for this deliverable: P3 UWE
Dissemination level: Public

Authors

Mr Andrew Branson
Dr Jetendr Shamdasani
Prof Richard McClatchey

Approval

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

History

	Date	Notes
Version 1	06 June 2012	Created Document
Version 2	01Aug 2012	Document populated with sections
Version 3	01 Oct 2012	Document populated with text
Version 4	15 Nov 2012	Document circulated internally to all partners for comments
Version 5	10 Dec 2012	Final version
Version 6	12 Dec 2012	CFc overall revision
Version 7	20 Dec 2012	Submission version

Table of Contents

Executive Summary

The purpose of Work Package 10 (*N4U Analysis Services*) is to develop a user environment for managing and executing scientific workflow executions on neuroimage datasets, known as the N4U *Analysis Service*. 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 thereby allowing users to share their experiments and results.

This document is deliverable 10.1 (*Specification of the User Analysis Environment*) due at month 18 of the N4U project. The purpose of this deliverable is to provide a specification of the Analysis Environment in the project. As well as this specification an evaluation of possible software is presented. Each of these pieces of software is a possible candidate as the basis of the Analysis Service. Also included in this document is the desired interaction with other N4U components and how the Analysis Service relates to these and a preliminary software delivery schedule.

Intended Audience

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

Note

This specification of the N4U Analysis Services is intended to directly address user requirements of the N4U project. As of the time of writing, the first deliverable in the project that will describe these requirements, D3.1: (*N4U Requirements for the Provision of Knowledge Management Services*) and Analysis Environment has yet to be approved so this specification may be revised if new requirements are received.

1. Introduction

This document is the specification for the Analysis Service, the core of WP 10 in the N4U project (*NeuGRID for you: expansion of neuGRID services and outreach to new user communities*). This service is a collection of user-space tools which utilize the Provenance, Persistency and Pipeline Services to specify and orchestrate the execution of scientific workflows for the analysis of clinical data, and to collect the provenance of the results. The Analysis Service is the primary user-facing element of the project, consequently the design of the WP10 services is driven directly by the user requirements (gathered in WP3, *User Requirements and Acceptance Studies*).

Section 2 introduces the N4U Analysis Service: it describes some desired requirements of the analysis services. Section 3 includes an evaluation of readily available analysis tools for scientific workflow tracking (not only limited to biomedical informatics). From this evaluation the CRISTAL system [1] is selected as the best choice for the N4U project, mostly due to the inherent provenance capabilities of CRISTAL which are discussed at length. An initial service integration design is discussed in Section 4. This design shows how the WP10 Analysis Service interacts with other services in the N4U architecture. Section 5 presents the N4U specific CRISTAL model which represents the information required to be captured by CRISTAL. A software release specification and schedule is described in Section 6. Finally, Section 7 concludes the document and states what the immediate future actions are for WP10 as a whole.

2. Introduction to the N4U Analysis Services

The purpose of the N4U Analysis Service is to provide a set of services which will expose a *Virtual Laboratory*. This is a platform which will be available for scientists to use the N4U gateway. Some features that the platform will provide are:

- The browsing of past analyses and their results.
- The creation of new analyses by pairing datasets with algorithms and pipelines found in the Analysis Base.
- The execution of analyses by creating jobs to be passed to the Pipeline Service, then logging the returned results in the analyses objects.
- Re-running of past analyses with different parameters or altered datasets and
- Analysis sharing.

The N4U Persistency Service stores clinical datasets in the Grid, and indexes their metadata in a database called the **Analysis Base**. This is the starting point for a user's interaction with the Analysis Service. To create a new analysis, the user must browse the Analysis Base to **decide which**

data they want to analyse, and **which analysis algorithm(s)** they want to be run on it. These choices essentially define the context for the user's analysis and need to be stored as a defining record of that analysis. Optionally, the user may decide to add post-processing work to the analysis to generate visualisations or summary reports after the algorithm has run.

When the specification of the analysis is complete, it can be run. This involves **sending jobs** out to the Grid for every element of the analysis algorithm selected, for every element of the dataset. This is done by creating a **child analysis** for each dataset element, with its own instance of the algorithm to be run, from which the Grid jobs are derived. This could potentially generate a large amount of parallel work, which the Grid will distribute to many computing elements.

As the elements of the child analysis complete, the Analysis Service keeps records of the state of the result set, which Grid resources were used for the computation, and the exact times the operation was started and completed as the **provenance** data of that analysis. If a pipeline was selected that requires more than one computational step to complete, or allows for a particular analysis to fork its work into more than one thread, then more jobs will be sent out until the pipeline is complete. Each of these are recorded, again as the provenance data of the analysis.

Once the child analyses of each dataset element are complete, any specified post-processing steps of the whole analysis can be performed in order to aggregate the results, generating tables and figures with popular software packages from the data produced by the pipelines, and the provenance of that result data. All of the data resulting from the analysis and its provenance is stored in the Analysis Base, where it enriches the dataset and algorithm profiles with **real-world usage data**.

An already complete analysis may be cloned, its parameters and subjects altered if desired, and run again. All analysis objects belong to, and are by default only visible to, the user who created them. It will be possible to share analyses with other users later in the project, and this functionality will be enhanced with more social-network style functionality in the final stages of the project.

3. Evaluation of Technologies

This section covers an evaluation of current technologies which have been considered to be used for the construction of the N4U Analysis Service in WP10. The technologies that have been evaluated are mostly workflow orchestration and provenance capture tools. This is because no tool is complete enough to provide an entire set of functionalities for the creation of the Analysis Service. Therefore, development has to be conducted on top of these tools in order to create a suitable analysis environment for the users of N4U. The tools evaluated in this section are Taverna [2], VisTrails [3], Pegasus [4], Kepler [5] and CRISTAL [1].

CRISTAL

CRISTAL is a Description Driven System (DDS) [6], it focuses on users creating workflows, custom models and scripts which are used to create a dynamic application. It is a very versatile piece of software which captures provenance by design. Provenance in CRISTAL is captured by creating events which are generatedat every step of the process, therefore nothing is ever deleted from CRISTAL. CRISTAL is able to create composite workflows, which means it is capable of embedding a complex workflow in a single activity.

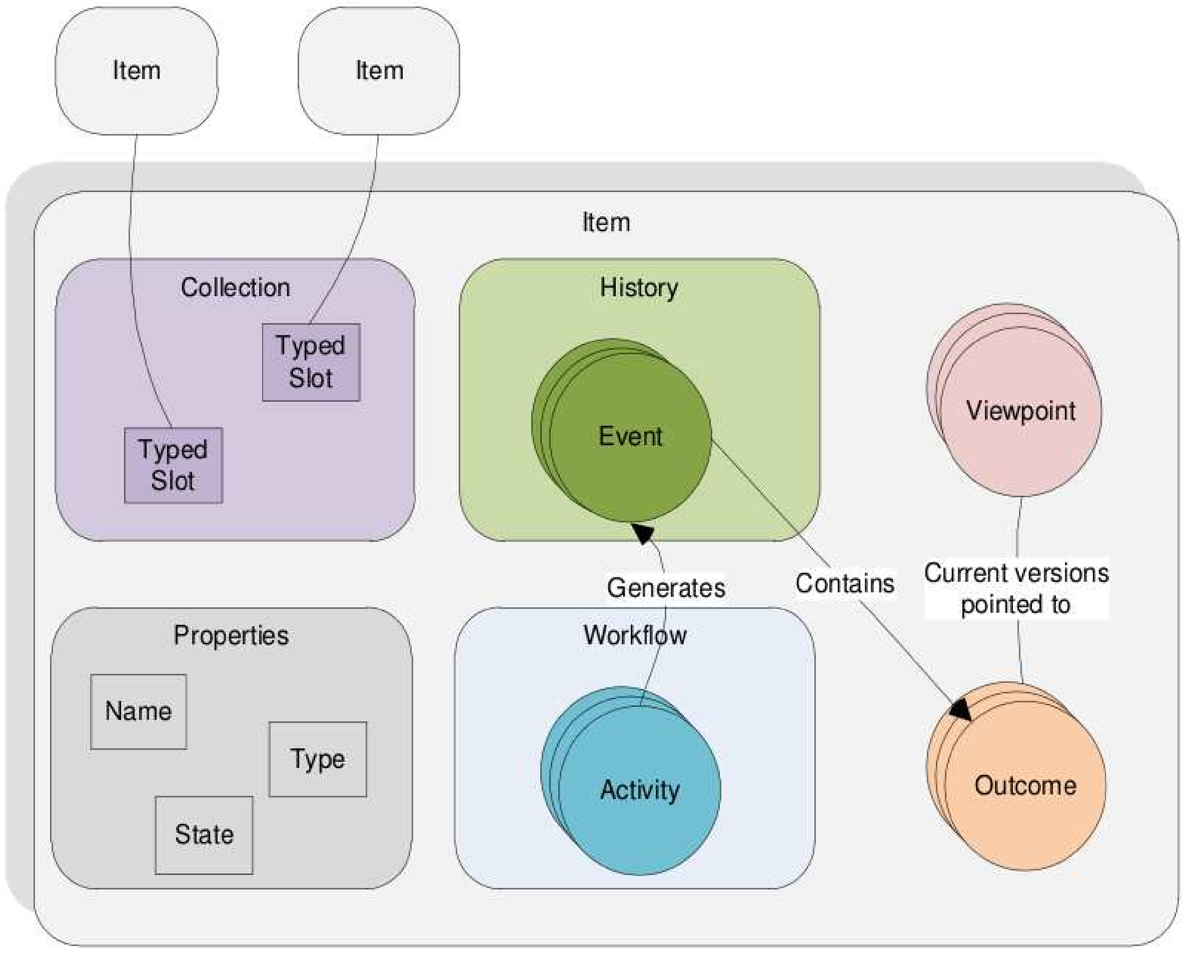

Figure 1: the internals of a CRISTAL Item.

Everything in CRISTAL is based on what is known as a *Item*. The current CRISTAL Item model is shown in figure 1. An Item contains the following:

- Workflows i.e. complete layouts of every action that can be performed on that item, connected in a directed graph that enforces the execution order of the constituent activities.

- Activities capture the parameters of each atomic execution step, defining what data is to be supplied and by whom. The execution is performed by agents.
- Agents are either human users or mechanical/computational agents (via an API), which then generate events.
- Events detail each change of state of an Activity. Completion events generate data, stored as outcomes. From the generation of an Event provenance information is stored.
- Outcomes are XML documents resulting from each execution (i.e. the data from completion Events), for which viewpoints arise.
- Viewpoints refer to particular versions of an Item's Outcome (e.g. the latest version or, in the case of descriptions, a particular version number).
- Properties are name/value pairs that name and type items. Properties also denormalize collected data for more efficient querying, and Collections enable items to be linked to each other

CRISTAL workflows are able to represent both processes and data flows. This makes them very powerful since they are able to represent anything that a user desires from CRISTAL. It is a mature system which has been used for the production of the CMS ECAL detector at CERN and is currently being used as the basis for commercialisation for a set of generic workflow based software products.

As stated previously CRISTAL captures provenance by design. In the ancestor to this project neuGRID [7], CRISTAL was used as a provenance capturing mechanism by being exposed as a service. The CRISTAL provenance mechanism functions as follows: an *Event* is generated by an *Agent* which alters a state in an *Activity*. The actual provenance information required is application dependant. In neuGRID CRISTAL was used to capture provenance of:

- Workflow specifications - These were XML based specifications of workflow descriptions which were external to CRISTAL. These were serialised and stored in an relational database.
- Data or inputs supplied to each workflow component - The parameters to each workflow component in the case of neuGrid and N4U these are images, however, they can be any piece of data.
- Annotations added to the workflow and individual workflow components - These consisted of simple name value pairs which allowed uses to store extra information about a workflow.
- Links and dependencies between workflow components - This is a part of the workflow specification.
- Execution errors generated during analysis - A necessary component for any provenance model.

- Output produced by the workflow and each workflow component - In the case of neuGRID and N4U these are images, however, as with inputs they can be any piece of data that the application requires.

Taverna

Taverna [2] is a workflow system the main users of which are from the bioinformatics community. Taverna has been freely available under an open source LGPL license and development of it has been encouraged due to its large user base. Workflows in Taverna are based on a dataflow model of computation as in they deal with transformations of data inputs to workflow vertices.
Taverna workflows consist of directed graphs, where the vertices, named *processors*, are either web services or local scripts. In Taverna, a processor node consumes data that arrives on its input ports and produces data on its output ports. Each edge in the workflow graph connects a pair of ports, and denotes a data dependency from the output port of the source processor to the input port of the sink processor. When data enters a processor it goes through what is known as a *dispatch stack*. Where various optimisations are performed, these are:

- Parallelise: this ensures that, when iterations over lists are involved, independent concurrent threads are created to process each list element;
- Error Bounce: is responsible for immediately terminating an execution when any of the inputs are in an error state. This protects the underlying activities from being invoked on invalid input;
- Failover: detects the failure of an activity associated to the processor, and is responsible for selecting an alternate activity, if available (recall that activities can be added and removed dynamically);
- Retry: provides tolerance to transient errors in the underlying activity, by repeatedly attempting the same invocation for a configurable number of times.

Workflow orchestration in Taverna conceptually proceeds by pushing data through the directed data links from one processor to all of its successors, starting with the items that are presented on the workflow inputs. A processor is ready to execute when all of its inputs ports are populated with a data item. Taverna uses the Freefluo workflow [8] enactor to enact web service processors. Taverna however being a very powerful workflow orchestration too, only considers provenance as an afterthought. State or data changes are not recording during the orchestration process. however, in CRISTAL they are recorded at every step of the workflow orchestration process.

The connections between activities in CRISTAL indicate the flow of control rather than data in their raw form. However these edges can also have domain specific properties attached to them which are visible to the receiving activities, making data flow an implementation choice at the activity executor level. This was how data-flow workflows were migrated into CRISTAL for neuGRID, although CRISTAL was not used as a workflow orchestrator in that project and thus no executor was ultimately implemented; this is, however, planned for N4U. Implementing data flow as a layer on top of control flow also means that it can be optional for parts of the system (such as those involved in workflow design) that do not require it, but the system still stands to gain from the collection of provenance.

VisTrails

VisTrails [3] is a system that enables the authoring of workflows, termed *vistrail specifications* that produce visualised outputs, termed a *vistrail*. These outputs may be, for instance, maps or brain visualisations. A vistrail specification consists of a sequence of operations, termed a *module*, used to generate a visualization. It records the visualization provenance which can later be used to automatically re-generate the images. A Vistrails specification is specified in an XML-based language and the execution of the workflow is orchestrated by the VisTrails Cache Manager. The VisTrails Player is used to visualise the output of the Vistrails specification. VisTrails supports cloud integration as well as workflow ensembles, termed *workflow medleys* where a workflow consists of other workflows. VisTrails is a specialized system only collecting provenance for its visualization process. For N4U we require a system which is able to collect provenance information for both data and process flows for when an experiment is executed, but we will consider the use of VisTrails in the WP10 N4U Visualization service.

Pegasus

Pegasus [4] is a workflow management system designed for scientific workflows. Unlike previous projects, but in common with CRISTAL, Pegasus implements a separation between abstract workflow specifications (workflow descriptions in CRISTAL) and concrete workflows (workflow instances in CRISTAL). Pegasus workflows are task-based directed acyclic graphs and are described in an XML based language called DAX. Wings for Pegasus [9] extends the DAX language for more rich semantic information about abstract workflows and for provenance handling. Although CRISTAL uses XML for persistency and communication, the format is not designed to be used as a programming language, but instead workflows are drawn using a graphical interface. It is possible to create CRISTAL workflows by transforming other XML formats, as was done in neuGRID.

Upon instantiation of a workflow in Pegasus, the abstract workflow specifications are mapped, via various transformations to a concrete executable workflow. This can then be executed in a Grid-based infrastructure, since CRISTAL workflows are instantiated from their descriptions, however the Pegasus abstract workflows are not themselves workflow driven, and thus cannot collect provenance unlike CRISTAL.

Recent work on Pegasus has focused on cloud-based enactment as well. Unlike CRISTAL, VisTrails and Taverna, a Pegasus environment is more deeply integrated with the underlying computing infrastructure. For instance, Pegasus requires information from the metadata catalogue of the Grid infrastructure to generate concrete workflows. Similarly, Pegasus requires information from the data replica catalogue of the infrastructure to perform data mappings in the workflow. These catalogues are analogous to the so-called Data Atlas of the N4U project, from which the users select the datasets and algorithms that will together instantiate a new analysis. Pegasus can implement various optimisations on the workflows before enactment. Workflow orchestration in Pegasus is handled by the Pegasus enactor. Pegasus also supports integration with external enactors such as DAGMan [10].

Kepler

Finally, Kepler [5] is a workflow management system designed for scientific workflows. Unlike Taverna, Vistrails or Pegasus, Kepler is an abstract workflow management system because it is designed to be flexible to support any form of workflow orchestration. It implements the actor/director model for workflow orchestration and description, again similarly to CRISTAL. Kepler workflows are described in MoML [11]. Actors (analogous to CRISTAL Agents) in Kepler denote specific tasks in a workflow, while directors orchestrate the execution of all actors to produce the final output (as does the CRISTAL server). An actor in a Kepler workflow may be a script, an executable task or a web service, which would be implemented in CRISTAL as domain specific agent processes. A director can orchestrate the execution of Kepler workflows in Grid-based environments [12], cloud-based environments or execute a workflow locally. The Kepler provenance framework handles the provenance information.

A fundamental difference between these different workflow systems and CRISTAL is that CRISTAL is designed with provenance in mind from the very beginning. Nothing can be changed in CRISTAL without an activity being executed somewhere, and every activity execution records the complete details of that execution in a replayable form. As previous states of the system are preserved intact, including the domain application design, it is possible to replay the entire system from installation up to the present. Other systems emulate this behaviour in external components and plug-ins, but this requires considerable work for this provenance to be complete to the same extent as with CRISTAL.

Image-J

Image-J [13] is 100% free software, it is written in Java and is developed actively. Its purpose is to provide a set of tools for image processing of scientific data. Image-J can manipulate 8, 16 and 32 bit colour images. It has a simple GUI which exposes all of its functionality and all of these functionalities are accessible through its Java API. It can read many image formats such as TIFF, GIF, JPEG, BMP, DICOM, FITS and 'raw'. It supports what are known as 'stacks' which are a series of images that share a single window. It is a multithreaded application so time consuming operations can be performed in parallel with more simple operations such as opening a file. Image-J was designed with an open architecture that provides extensibility of its functionalities via a plugins written in Java. User-written plugins allows Image-J to solve any image processing problem. We in WP10 feel that this is a suitable tool to use in the WP10 Analysis Service for visualization purposes. This is because the API is 100% free and open, therefore, it can be wrapped up as a service within the WP10 framework.

4. WP10 Service Integration Model

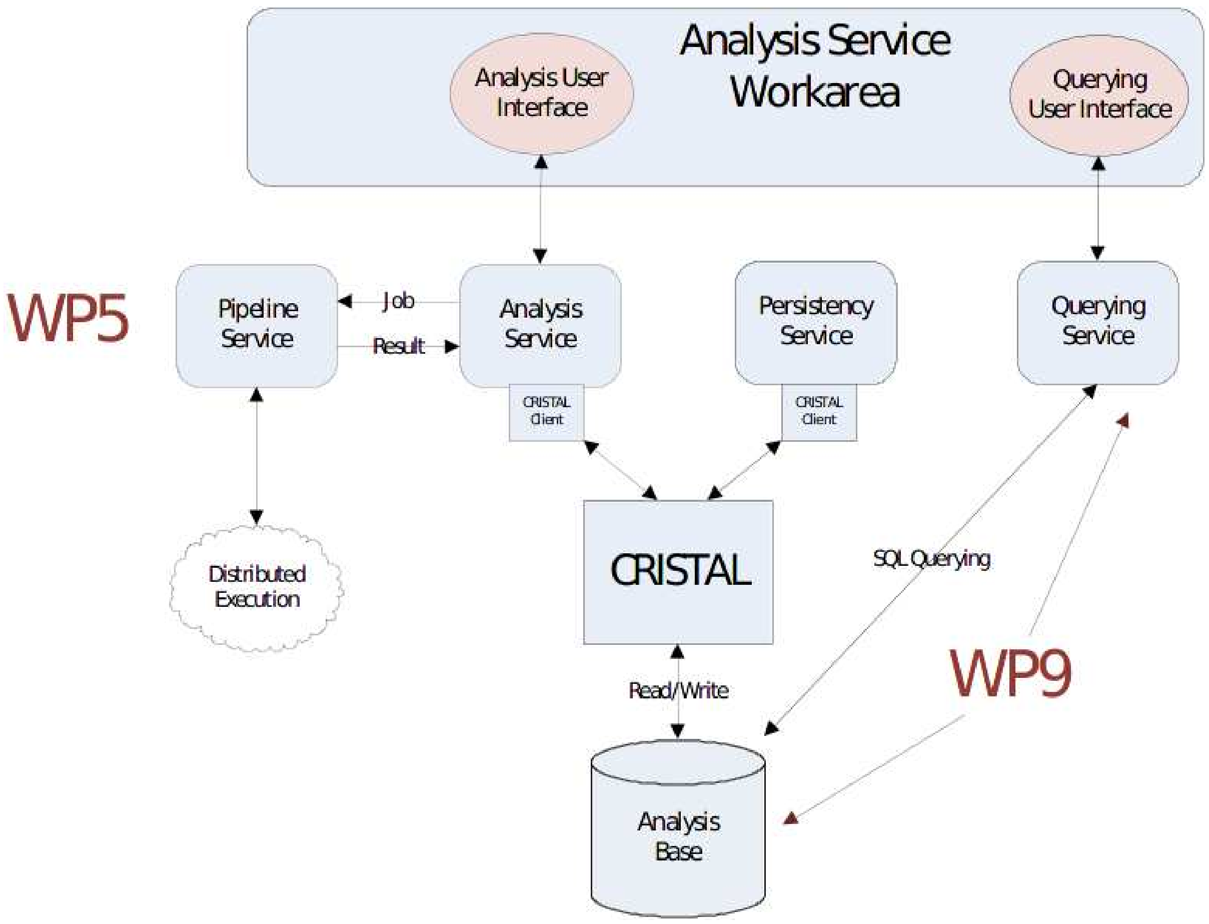

Figure 2: WP10 Service Architecture.

WP 10 will deliver a set of services to be used by clinicians in the N4U project. Fig. 2 above shows how WP10 intends to sit with in relation to the N4U architecture. Here CRISTAL is a nucleus and driving force between N4U services. It will integrate with both WP5 *DCI Extension Services Provision* (the pipeline service) and WP9 (the analysis base). The components are the following:

Pipeline Service - This is the Pipeline service from WP5, that was originally written as part of the neuGRID project. Its purpose is to submit jobs to a distributed execution environment (in the case of N4U it is the Grid) and to retrieve the results. It has been heavily modified already since neuGRID due to changing technologies, and is planned to support a workflow engine in the near future. Although there may be potential to submit workflow fragments for closely related jobs to it in the future, to start with the analysis service will only send single jobs.

Analysis Service - This service is used to expose CRISTAL functionality to the N4U platform. This means that it is the main interaction point users to interact with CRISTAL, to manage pipeline specifications and instantiate new analyses. It will submit individual jobs to the pipeline service and retrieve the result. This will connect to the CRISTAL server. The Analysis User Interface component also uses this service as an entry point to CRISTAL.

Persistency Service - This service is from WP9 (*N4U Information Services*). Its purpose is to import data into the analysis base. It currently connects to CRISTAL after formatting the datasets to a application specific format. CRISTAL then performs a write to the Analysis Base directly.

Querying Service - This is from WP9 once more and used for data browsing and data access. It does not write directly to the Analysis Base, all writes must go through CRISTAL. Analysis Base - This is the main index and data store for N4U. This is a part of WP9 and the schema will be described in their deliverable. The key point to note is that all writing to the Analysis Base is done through CRISTAL exclusively.

The Analysis Service Work Area - In this design this currently contains two specific user interfaces. These are the Analysis User Interface and Querying User Interfaces. In the original DOW there is a visualisation service present. However, we have now integrated this into the Analysis User Interface since it is strictly a user facing component so that clinicians can make sense of their data.

The Analysis Base - This is the database that sits behind CRISTAL. The only means of writing to this database is through CRISTAL. The Analysis Base is an index of data and is exposed to users by the querying and browsing of this indexed data through CRISTAL and the Querying Service developed by WP9.

Example Usage of the WP10 Services

This subsection details how the clinician *might* use the service architecture shown in Fig. 2, specifically in the scenario of creating a new analysis for their experiment. When a clinician sits down at her desk and wishes to conduct a new analysis, her first step would be to compile a selection of data from the datasets which are available to her. To do this she would log into the

Analysis Service Area and interact with the Querying Service through its user interface to find data that possesses the particular properties she is looking for. She submits her constraints, which are passed as a query to the Querying Service. The Querying Service then queries the Analysis Base which would return a list of dataset properties and locations which meet her constraints. The Querying Service interface would then display this list to the clinician to approve.
Once the user is satisfied with her dataset selection she combines it with a pipeline specification to create her Analysis. To do this she would need to use the Analysis Service Interface to search CRISTAL for existing *algorithms* that she can use to create a new *pipeline* or select a pre-defined pipeline. An analysis is an instantiations of a pipeline in the context of dataset and a pipeline. Command line utilities will be provided to aid in the creation of a pipeline by connecting different algorithms together as steps. The completed pipeline will have a dataset associated with it. Once this pipeline is ready it will be run on each element of the dataset by CRISTAL.

The pipeline will be sent to CRISTAL which will orchestrate the input pipeline. Currently the pipeline service from WP5 is not able to perform workflow orchestration. Therefore a single activity from the input workflow will be sent to the pipeline service as a single job using the pipeline API. Once the job as completed, the result will be returned to CRISTAL. Here CRISTAL will extract and store provenance information for this job in the analysis. This information will contain factors such as the time taken for execution, and whether the job completed successfully. It will store this information internally in its own data model. It will also post this information to the Analysis Base so that this crucial provenance information is accessible by the Querying Service. This loop of sending jobs and receiving the result will continue until the workflow is complete. Once this workflow has completed CRISTAL will once more generate provenance information and store this provenance for the entire workflow in its own internal data store and the Analysis Base. The final result of the completed workflow/pipeline will be presented to the user for evaluation. A link to the completed result in the form for a LFN (a GRID location) will be stored in the Analysis Base.

5. N4U CRISTAL Model

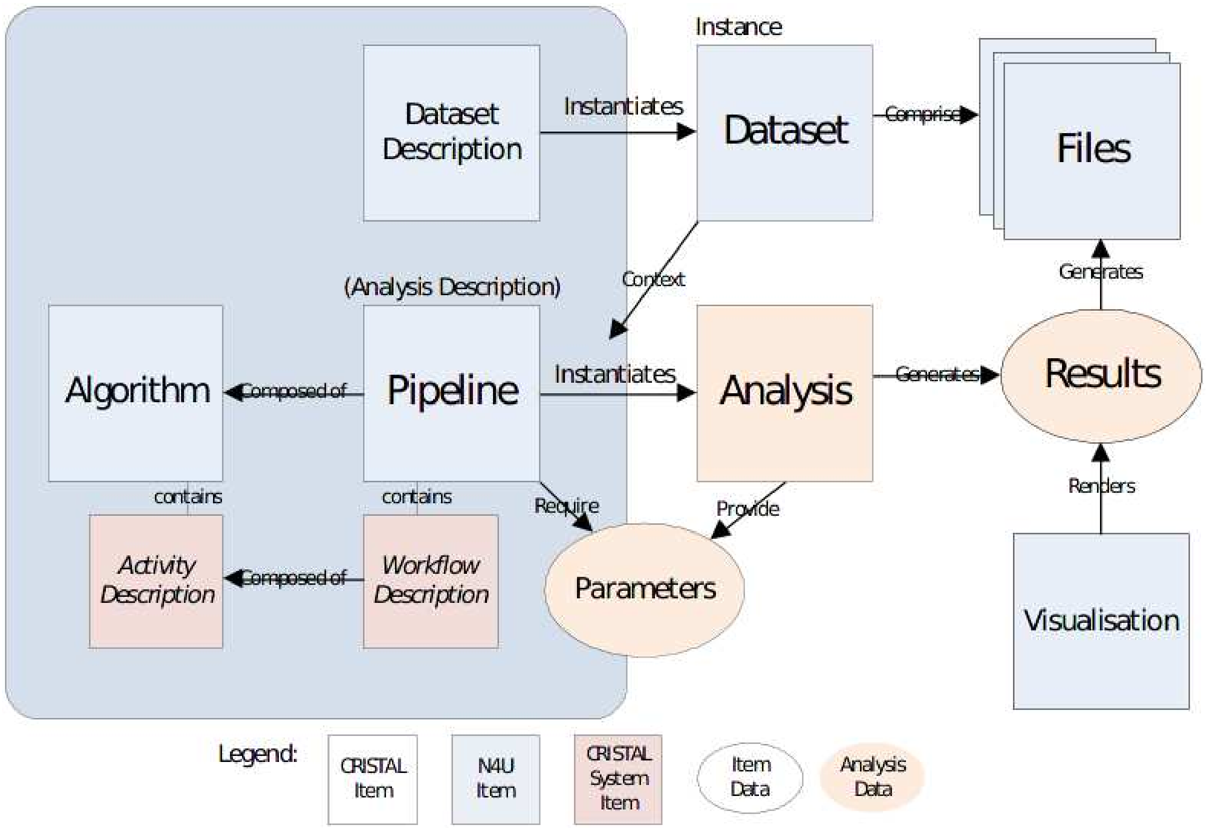

Figure 3: the N4U CRISTAL Model.

Fig. 3 shows the current design for the CRISTAL N4U Model. This model is an *internal* CRISTAL model which has yet to be instantiated. The squares in Fig. 3 represent CRISTAL Items and the ovals represent Item Data. There are two types of CRISTAL Items present in Figure 3 these are N4U Items (blue) and CRISTAL System Items (pink). There is only one type of Item Data present this is Analysis Data (orange).

The N4U Items are the following :

A *Dataset Description* is an XML based description of a *Dataset* object; which is an *instance of* a *Dataset Description* which contains; *Files* which are representations of image files; and finally a *Pipeine* (Workflow) is composed of *Algorithms*. The *Visualisation* Item renders the results from the Files.

There are two types of CRISTAL System Items these are an *Activity Description* which is the description for an *Algorithm*. A group Activity Descriptions make a *Workflow Description* which are the descriptions for *Pipelines*. These are XML descriptions which are later instantiated.

The Analysis Data objects are: *Parameters* which are objects that parameterise Pipelines. These are provided by an *Analysis* which is a collection of Pipelines. A set of *Results* are generated by the final *Analysis*.

6. Release Specification

The releases for WP10 in N4U are done in three stages these are the *Alpha*, *Beta*, and *Final* releases. The initial Alpha release will be delivered by month 24 of the project, the Beta by month 36 and the final version of the WP10 Analysis Environment will be available from month 42. The Alpha version is a prototype with simple functionality for demonstration purposes. It will be able to submit a single workflow with one job to the pipeline service which is a single algorithm that runs on a single image. After this job has finished execution the result from the Pipeline Service, it is sent back to CRISTAL. CRISTAL then creates the provenance information for this single task and stores it internally. After this process is completed, CRISTAL then connects to the Analysis Base (WP9) where it stores the relevant metadata associated with this execution. This initial alpha version will provide a simple API and command line interface for users to exploit.

The Beta version is a functionally near-complete version of the system for testing. The Beta version will provide a set of command line utilities and an expanded API from the previous Alpha version. It will be a complete instantiation of the diagram of services in Fig. 2, with all the services functioning and implemented. It will support initial creation of pipelines (workflows) and submit them to the N4U infrastructure. This version will also allow a user to run their pipelines on any dataset that they have uploaded to the infrastructure. Documentation for users will be released at this stage, this will guide users on how to use the system as it currently stands. Since this is a Beta version it is not final, but is released for user testing and evaluation. Users will be able to comment and submit bugs concerning the system at this stage. The final version of the software will be delivered at month 42. This will be the mostly bug free version of the N4U Analysis Suite and will be considered to be production ready.

7. Conclusions and Future Plans

This deliverable has presented the work done in WP10 thus far. We have evaluated various technologies (CRISTAL, Taverna, Vistrails, Pegasus and Kepler). CRISTAL from our comparative analysis was the clear winner since it provides provenance from the outset. The capture of provenance is an essential requirement from N4U so that clinicians can recreate past experiments. Other systems were not able to provide the provenance at such a fine grained level as CRISTAL.

As an addition to the previously presented evaluation this document has shown the initial design of services and how they interact with each other. This shows how central the role of CRISTAL is within WP10. An initial model has also been presented which captures the N4U items which need to be stored. In the next step this model will be instantiated within CRISTAL. This means that an N4U module will be created which will extend the CRISTAL kernel providing functionality that is N4U specific, this is the next immediate task of WP10.

8. References

[1] McClatchey R et al. *A Distributed Workflow & Product Data Management Application for the Construction of Large Scale Scientific Apparatus.* NATO ASI Series F: Computer & Systems Sciences. Springer Verlag. 1998; 164: 18-34.

[2] Missier P et al., Taverna, Reloaded. Proc of the 22nd Scientific and Statistical Database Management Conference (SSDBM). pp471-481. Heidelberg, Germany, 2010. And http://www.taverna.org.uk/ last accessed Jul 2012.

[3] Scheidegger C et al. *Querying and Re-Using Workflows with VisTrails*. In Proc of ACM SIGMOD Int Conf on Management of Data, pp 1251-1254, 2008. And http://www.vistrails.org/, last accessed May 2012.

[4] Deelman E et al., *Pegasus : Mapping Scientific Workflows onto the Grid*. Across Grids Conference 2004, Nicosia, Cyprus. And http://pegasus.isi.edu/ last accessed Oct 2012.

[5] Ludascher B et al. *Scientific Workflow Management and the Kepler System*. Concurrency and Computation: Practice & Experience. August 2006 Vol 18 (10):1039-1065

[6] Estrella F et al., Pattern Reification as the Basis for Description-Driven Systems. J Software & System Modelling. 2003; 2(2): 108-19.

[7] Redolfi A et al. *Grid infrastructures for computational neuroscience: the neuGRID example.* Future Neurol. 2009;4 (20):703-22.

[8] Freefluo Engine see http://www.taverna.org.uk/developers/taverna-1-7-x/architecture/freefluo-engine/, last accessed July 2012.

[9] Simmhan Y et al., *Performance Evaluation of the Karma Provenance Framework for Scientific Workflows*. Proc of the 2006 Int Conf on Provenance and Annotation of Data (IPAW'06). 2006; 222-36, 2006. And http://d2i.indiana.edu/provenance_karma.

[10] DAGMan - Directed Acyclic Graph Manager. See http://research.cs.wisc.edu/condor/dagman/, last accessed Nov 2012.

[11] Lee E & Neuendorffer S. *MoML: An Modeling Markup Language in XML*. Technical Memorandum ERL/UCB M 00/12. Available from Citeseer : 10.1.1.3.2696.pdf, last accessed Oct 2012.

[12] Ludascher B et al. *Scientific Workflow Management and the Kepler System*. Concurrency and Computation: Practice & Experience. 2006;18 (10):1039-65.

[13] Image-J http://rsbweb.nih.gov/ij/index.html last accessed Nov 2012