Grant agreement no. 211714

neuGRID

A GRID-BASED e-INFRASTRUCTURE FOR DATA ARCHIVING/ COMMUNICATION AND COMPUTATIONALLY INTENSIVE APPLICATIONS IN THE MEDICAL SCIENCES

Combination of Collaborative Project and Coordination and Support Action

Objective INFRA-2007-1.2.2 - Deployment of e-Infrastructures for scientific communities

Deliverable reference number and title: **D10.1 Gridification Model Specification**

Due date of deliverable: **Month 12**

Actual submission date: **31st January 2009**

Start date of project: **February 1st 2008** Duration: **36 months**

Organisation name of lead contractor for this deliverable: **maat Gknowledge**

Revision: Version **1**

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

Glossary

Term	Definition
IGT	Infrastructure Ground Truth. Level 0 of neuGRID's infrastructure hosting the GCC and DCC sites and associated test-bed
GCC	Grid Coordination Center. Core (common to all sites) services of the grid infrastructure
DCC	Data Coordination Center. Core (common to all sites) services of the database infrastructure
DACS	Data Archiving and Computational Site. neuGRID site offering and managering a set of physical resources
DCS	Data Collection Site. End-user sites acquiring data and connecting to a given DACS
Gridification	The engineering process of porting an existing application to the grid, so that it can be executed via the grid enactment environment
Pipeline	A pipeline is a set of data processing elements connected in series, so that the output of one element is the input of the next one (extracted from Wikipedia.org)
Workflow	A workflow is a depiction of a sequence of operations, declared as work of a person, work of a simple or complex mechanism, work of a group of persons, work of an organization of staff, or machines (extracted from Wikipedia.org)
Imaging Algorithm	An application, typically under the form of a Unix-like binary which manipulates imaging data
LORIS	The databasing software used in neuGRID, which offers interfaces to acquire and quality control data
Cortical Thickness	The key concept behind the structure of the cerebral cortex is its thickness. The goal is to measure the distance between the white matter surface and the grey matter surface across the entire cortex in order to analyse regional variations within individuals, and more importantly, across subjects
Healthgrid	A grid-based environment in which data of medical interest can be stored and made easily available to different actors in healthcare systems such as physicians, healthcare centres, patients and citizens'
SOA	Services Oriented Architecture
SOMA	Services Oriented Modelling and Architecture
VUmc	VU Medisch Centrum
KI	Karolinska Institute
FBF	Fatebenefratelli
PACS	Picture Archiving and Communication System
Voxel	A voxel (a combination of the words volumetric and pixel) is a volume element, representing a value on a regular grid in three dimensional space. This is analogous to a pixel, which represents 2D image data.
MINC	Medical Imaging NetCDF
NetCDF	Network Common Data Form
DICOM	Digital Imaging and COmmunications in Medicine
Analyse	Analyze is an image processing program and data format, written by The Biomedical Imaging Resource at the Mayo Foundation
GPL	General Public Licence
WSDL	Web Service Description Language
W3C	World Wide Web Consortium

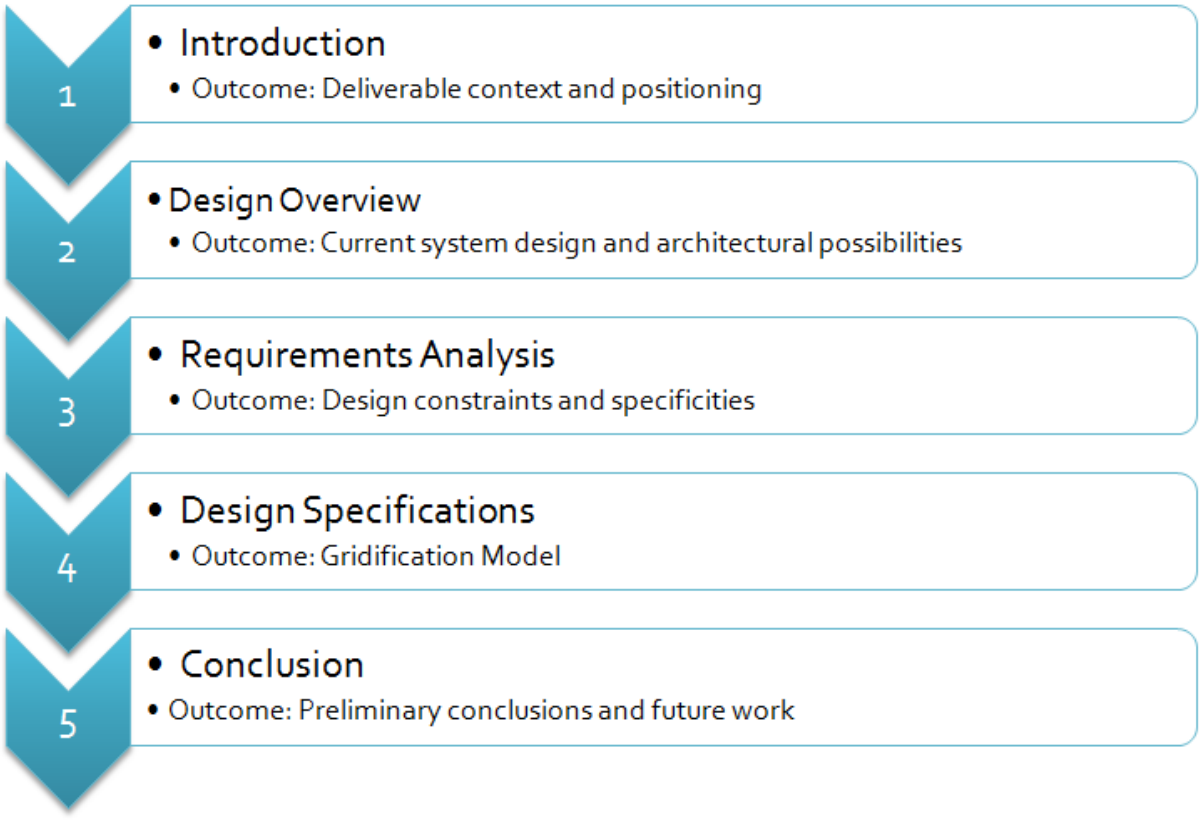
*Note: glossary terms are followed by * symbol in the remainder of this document.*

Executive Summary

The present document attempts to sketch the neuGRID gridification model and alternatives to be used for distributing the processing of neuro-scientific pipelines with a concrete application and test of the neuro-imaging Cortical Thickness* pipeline (see work package 5 – WP5 – for more information) in the grid infrastructure, expected to take place at the end of year 2 as a validation case study. Beyond the domain specific application and context of specified models, the presented design aims to be as generic as possible in order to eventually accommodate with other medical fields on the long run.

This first version of the document follows/ accompanies the user and system requirements analysis process, while delivering early insights on possible solutions. It is indeed anticipated that the document will know a series of refinement iterations in light of the requirements analysis delivery (at Month 14) and all along the neuGRID platform development.

The document is structured as follows:



Readers should therefore be aware of the preliminary content nature and its expected subsequent developments. As such, work package 10 (WP10) will deliver an update at Month 24 following the deliverable "*D10.2 Gridified Toolbox Portfolio Report*", which will confirm/ impact on some of the solutions hereinafter described.

1. Introduction

This active document is meant to be refined all along the neuGRID project lifetime. It aims to provide a set of high-level design specifications useful to support and structure the developments of the gridification components associated to the WP10, which are in line with the requirement specifications being produced as part of the analysis carried out in work package 9 (WP9). Most specifications within the document are expressed using ad-hoc representations, for which explanations are provided where and when judged appropriate.

1.1. Purpose of the Document

This document illustrates the design approach and specifications of WP10, processed during the first 12 months of the project, which have supported and driven the prototyping efforts. This work has been carried out within the task entitled "*T10.1 Algorithms Elicitation & Gridification Models*", which started at Month 7 and completed at Month 10, with the following objective (extract from the project description of work):

"Conceptual integration of the different system components developed in WP5/6/7. Evaluate the existing algorithms in terms of software and hardware dependencies, computing and storage requirements, and define the level of gridification to be applied. Define a gridification model and scheduling policy covering all concerned algorithms and expandable to new ones (P4 MAAT, P2 NE, and P3 UWE). A document will be produced by P4 MAAT, P2 NE and P3 UWE illustrating the adopted approach and resulting implementation of the gridification mechanism (D10.1) (M12)."

In the following sections, the reader will gain understanding on the approach that is driving the work package and its developments. It establishes the features which the solution should support, and as such, it is intended to be a useful reference throughout the development and testing phase of the neuGRID prototype.

1.2. Document Positioning and Intended Audience

WP10 "*Algorithms and Pipeline Gridification*" aims to make existing brain image analysis algorithms compatible with the grid environment (the so-called process of "gridification") and to develop the necessary foundations for their publication, management, sharing, combination, and scheduling in the neuGRID system. The outcome of this work package will be an algorithm "*toolbox"* made available to end-user communities through the platform, which can be discovered, enriched, invoked and applied in different ways onto a large set of images and clinical records.

More precisely, the work package aims to (extract from the project description of work) *(1) evaluate the existing algorithms' implementations and requirements in terms of software adaptation and interfacing, (2) design and implement a set of distributed and cooperative optimization methods for facilitating algorithms gridification and their future scheduling within the platform, (3) design and implement a set of interfaces for managing the algorithms in the grid (from algorithm publication, to versioning, to training, to sharing), (4) gridify the algorithms, re-engineer algorithms inner interfaces so that their input/output can be plugged in the grid, (5) define adapted scheduling policies for the selected algorithms, based on algorithms requirements, (6) design and develop a grid-based workflow management system for combining, optimizing, executing and monitoring algorithms, which extends the workflow capabilities of the grid middleware and addresses fully the brain image processing algorithms needs, and last but not least (7) extend the scheduling possibilities.*

Thus, the presented specifications are destined to serve in priority all protagonists of the Joint Research (JRA) and Services (SA) activities of the project, more particularly to IT researchers, IT developers and neuro-scientific algorithm developers involved in the following work packages:

Services Activities – SA

WP Id	*WP Title*	*WP10 Contribution*
WP5	Brain Imaging Services Provision	To provide a gridification model specification and corresponding implementation for brain imaging services to be published, discovered and executed in neuGRID's platform
WP6	Distributed Medical Services Provision	To provide a Workflow/ Pipelining service and corresponding API for higher level distributed medical services to submit, execute and monitor algorithms/ pipelines
WP7	Grid Services Provision	To dictate the deployment of necessary underlying grid services and corresponding configurations
WP8	Deployment Services Provision	To dictate the deployment of necessary underlying neuGRID services and corresponding configurations

Joint Research Activities – JRA

WP Id	*WP Title*	*WP Relation*
WP9	User and System Requirements Analysis	To conform with requirements analysis conclusions
WP11	Platform Integration, Performance and Feasibility Tests	To support neuGRID services integration

To a lesser extent, since indirectly concerned (through the natural abstraction of Workflow/ Pipeline authoring environments such as the ones proposed in WP6, see D6.1 for more details), the Neuro-Scientists and prospective users (e.g. Pharmaceutical industries) as well as inside and outside reviewers of the project activities, are anticipated as potential additional audience.

1.3. Reference Documents

Prior to reading this document the reader should be familiar with additional documents/ deliverables produced within the neuGRID project, which have or are considered to potentially impact on the design and developments of WP10. The following is a list of such documents sorted by information sources, activities and corresponding work packages (Note: list of available documents at the time of writing):

Services Activities Related Documents

WP Id	*WP Title*	*Documents*
WP5	Brain Imaging Services Provision	D5.1. Brain Imaging Service Portfolio Specification Document
WP6	Distributed Medical Services Provision	D6.1. Design Document including API Documentation and Description of Functionality for the Underlying Layer
WP7	Grid Services Provision	D7.1. Test-bed Installation and API Documentation

WP8	Deployment Services Provision	D8.1. Ground Truth and Phase 1 Deployment Test and Validation Report

Joint Research Activities Related Documents

WP Id	*WP Title*	*Documents*
WP11	Platform Integration, Performance and Feasibility Tests	D11.1. AC/DC1 and Story Lines Test Suite Specification and Report neuGRID Architectural Considerations Presentation (see neuGRID CMS, WP11 directory: https://www.neugrid.eu/owl-0.90)

Other Related Documents

Title	*Documents*
Project Documents	Project Description of Work
Requirements Analysis Supporting Material	Workflow related requirements analysis material, including recordings, meeting minutes and pipelines examples (see neuGRID CMS, WP10 directory: https://www.neugrid.eu/owl-0.90)

2. Design Overview

2.1. Objectives/ Quality

As the present document is destined to serve as a reference throughout the WP10 developments, it aims to satisfy a number of general design objectives:

- Develop a coherent representation of software that will satisfy requirements expressed here and greater detailed in deliverable D9.1. It gives a preliminary structure to developments, to integration of external contributions and recalls the design objectives (to be) pursued in WP10. As such, developing a coherent representation of the software implies actions to be carried out to:
    - Decompose the system into sub-systems that provide related sets of services/ components, enable the separation of concerns, and help in better splitting the work among partners,
    - Establish a framework for sub-system control and communication, useful for harmonizing and gluing the potentially heterogeneous resulting components/ technologies,
- Identify inadequacies in requirements, as early as possible in the development process, supporting partners to make appropriate decisions over time,
- Provide a reference tool readable by developers, testers, and maintainers. Beyond formalizing what the system should do, the document also serves as a conceptual map that actors of the project can consult at anytime to better understand/ locate/ solve technical issues,
- Provide a basis for integration and testing.

As such, the document also attempts to exhibit several qualities:

- Complete: everything that is essential is described. All WP10 system blocks to be implemented during the project are specified and placed appropriately within the system architecture,
    - Rigorous: expressed in a well-defined notation. Diagrams are formalized, as much as possible, following standard notations when possible,
    - Uniform: the entire document is at the same level of detail and remains an abstract description of the targeted system,
- Desensitized to change: it voluntarily hides implementation details to remain a high level specification,
- Modifiable: this document will change over time. As expressed earlier, the presented specifications will be revised as requirements analyses progress. Thus, it is intended to be updated at Month 24.
- Confirmable, verifiable and testable, the resulting prototype system should illustrate the presented specifications. In case of significant deviations, such specifications will be revised to better understand the mis-matches and their origins, with the aim of re-aligning both design and developments.

This document addressing the difficult task of shaping up the WP10 design specifications at a sufficiently high level of abstraction, it will not cover the following aspects:

- Technical Specifications of the prototype system. Indeed, no technologies adopted in the prototyping phase will be described nor analyzed in the following sections,
- Prototype. The approach undertaken for developing the system as well as the temporarily adopted technologies will not be discussed.

2.2. Brainstorming Meetings

In order to develop a common language and to seek technical agreement, the neuGRID project partners have conveyed a number of requirements gathering and technical brainstorming meetings. The following is a list of major held meetings with corresponding objectives/ achievements over the first year of activity in the project:

Date	Location	Content
2008-02-04	Fatebenefratelli – Brescia, Italy	Project kick-start. Initial series of requirements meetings and technical brainstorming on fundamentals of neuGRID system
2008-03-15	Karolinska Institute – Stockholm, Sweden	Second series of requirements meetings
2008-03-26	Teleconference	Initial Project Management Team (PMT) Meeting and thus technical brainstorming
2008-04-21	Teleconference	PMT Meeting and technical discussion
2008-05-15	VUmc – Amsterdam, The Netherlands	Third series of requirements meetings and in person technical brainstorming
2008-07-17	Teleconference	PMT Meeting and technical discussion
2008-07-21	Teleconference	First Services Area (SA) Teleconference and corresponding technical discussion
2008-07-31	Teleconference	First WP10 teleconference and technical discussion related to Cortical Thickness pipeline requirements
2008-10-10	Teleconference	Second SA Teleconference. Presentation of candidate system architecture
2008-10-17	Teleconference	First JRA Teleconference. Follow-up to SA teleconference discussions with elements of JRA
2008-11-05	Teleconference	PMT Meeting and technical discussion
2008-12-01	CERN – Geneva, Switzerland	In person technical brainstorming meeting. Validation of gross system architecture and responsibilities split between work packages. Agreement on LORIS integration scheme
2009-01-19	Teleconference	PMT Meeting and technical discussion
2009-01-22	PRODEMA – Montreal, Canada	In person technical brainstorming meeting between MAAT, UWE and PRODEMA. Agreement on technical roadmap for LORIS integration and grid connection

From this list, it can be noted that all technical partners have participated actively in the design process thus allowing the collaboration to make reasonable progress over the first year. The six first months of requirements analysis activity have helped greatly at scoping the design and precisely understanding the complexity related to imaging pipelines gridification, although further testing and prototyping is now required following which an update of the document will be produced.

The major milestone for WP10 over the period has been the system architecture gross design consensus at Month 9, which has given an accurate framework for the gridification model specification, as detailed in subsequent sections of this document.

The remainder of this deliverable attempts to formalize part of the thus far gathered requirements with a special emphasis on workflow/ pipelines of neuro-imaging/ data-mining algorithms and corresponding possible gridification model.

2.3. Rational

2.3.1. Approach to Design

The major goal that guided the design specification process throughout was to establish a common coarse-grained view of the system in light of the freshly gathered requirements, useful to identify major layers, inner constituents and corresponding interfaces, as well as to help in better splitting the work and responsibilities among the work package members.

Thus, WP11 – in charge of the project platform integration and tests - kick-started the design of an overall and gross architecture specifying logical layers grouping system functionality per areas. Similarly to the Service Oriented Modelling and Architecture (SOMA) [1] process, partners went through the exercise of identifying features and gradually grouping them into layers, to then specify and implement them.

At the confluences of requirements analysis – following a top-down elicitation process – and underlying bottom-up grid deployments, WP6, WP10 and WP11 have undertaken kind of a meet-in-the-middle approach to align system architecture with end-users' expectations. The result has undergone formalization in different documents, respectively *D6.1 "Design Specifications of Generic Medical Services", D9.1 "User and System Requirements Analysis" and D10.1 "Gridification Model"* using a Services Oriented Architecture (SOA) [2] as the focal meeting point and federating concept.

As a consequence and to give clarity to this deliverable, only a relevant subset of the resulting requirements and design objectives is presented, with the aim of covering WP10 gridification related components. The following section thus briefly presents the service orientation and associated advantages; aspects which then introduce the retained system architecture and give clear positioning of WP10's contribution.

2.3.2. Service Orientation

The main characteristics of an SOA are the loose coupling between services, the abstraction from technological aspects and its extensibility; features considered essential to cope with distributed developments, heterogeneous technologies integration and to leverage multi-partners collaborations.

SOA provides a simple yet efficient way to reuse software artefacts through the concept of standard services that are not bound to each other. Technological abstraction is obtained from using service contracts that are platform-independent. Extensibility is finally reached through service discovery and composition at execution time. Several definitions of the concept can be found in the literature, however for the remainder of this document, only the three following are retained, as they are most relevant to this work:

- "*A service-oriented architecture is a style of multi-tier computing that helps organizations share logic and data among multiple applications and usage modes*" as stated in 1996 by the Gartner group.

- "*Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations*" established in the OASIS reference model.

- "*SOA enables flexible integration of applications and resources by: (1) representing every application or resource as a service with standardized interface, (2) enabling the service to exchange structured information (messages, documents, "business objects"), and (3) coordinating and mediating between the services to ensure they can be invoked, used and changed effectively*".

In spite of the absence of a single officially and consensually agreed definition for SOA, three key roles are usually identified: service producers, service brokers and service consumers. The service producer's role is to deploy a service on a server and to generate the description of this service (i.e. so-called the service contract), which defines available operations as well as invocation mode(s). This description is published in a directory of services inside a service broker. Thus, services consumers are able to discover available services and to obtain their description by interacting with the service directory. The obtained descriptions can then be used to establish a connection with the producer and to invoke the desired service operation(s).

Just as for the SOA concept, loose coupling does not benefit from a unique definition. The commonly adopted approach though is to introduce a minimum of dependencies between services in order to better support their reusability. Moreover, these services should be combined in order to quickly and cost-efficiently respond to new demands. To achieve this goal, some engineering rules which are not always specific to SOA, have been identified [3]. Encapsulation and abstraction principles originally came from the world of object-orientation. The idea was to hide self-contained information of a service to end-users and to propose only one stable interface stressing the details considered to be necessary for handling it. A service is therefore seen as a black box from the outside, which makes it possible to separate its interface (i.e. its external description) from its actual implementation. One can thus modify a service implementation without changing its interface, which turns it into a sustainable model. The following rules are more specific to SOAs:

- (A) A simple and ubiquitous interface must be provided by any service and must be universally accessible by all suppliers and all customers of services. Thanks to a generic interface, it is then possible to interconnect any services and to forward any messages between the various interfaces. The keyword here is "decoupling" and it can take various roles: (1) to reduce the coupling between modules, for improved reusability, (2) to reduce the coupling with respect to the infrastructure and to the implementation platform, for improved interoperability and (3) to reduce the coupling between a service consumer and a specific implementation of this service, for improved evolution. In Web service architectures, the consensus to achieve this rule is to use Web Service Description Language (WSDL*).

- (B) Messages delivered by a service should not contain business logic. On the contrary, they must be restricted to the transport of, and only of, data structures from one service to another. That makes it possible to modify or to add services without impacting the other services of the architecture. These data structures can nevertheless be very complex in order to deal with security management (i.e. authentication, encryption, authorization, etc) or even file transfer.

These aspects are addressed thanks to different specifications that strengthen the "standard" Web service architecture (e.g. WS-Security, SOAP-attachments, etc).

- (C) A well-formed service must be stateless. This rule, which can seem very constraining, must be moderated though. It is recommended that the state conservation (i.e. the management of the context) as well as the action coordination (i.e. the management of the transactions) are localised in a specific function of the SOA, such as the orchestration. The application of such a rule facilitates the reuse, the scalability and the robustness of services and thus resulting SOA. Moreover, this rule enforces the loose coupling.

- (D) Cohesion is a difficult rule to define. It translates the degree of operations and functional proximity inside a service. In other words, it aims at facilitating the comprehension and reusability of a service by gathering homogeneous operations belonging to the same functional area.

- (E) A service should be idempotent. That makes it possible to be unaware of multiple receptions of the same request. The idea is that the use of such a service makes it possible to slacken the assumptions of reliability on the communication layer. In Web service architecture, the WS-Addressing specification allows among other things to enforce part of this rule.

If some of these rules can or sometimes should be moderated according to system requirements, i.e. the stateless and the idempotent ones, all these recommendations remain vital to create an open, sustainable and standard SOA. Indeed, these characteristics are mandatory in order to cope with heterogeneous resources ranging from data, to knowledge, to applications, and beyond software, to people. The SOA approach makes it possible for a wide range of collaborators having different skills/ backgrounds to develop together a system extensible to different application areas, which is of absolute importance in the case of neuGRID.

Aiming at addressing these challenges, the neuGRID partners have therefore started complementing the grid middleware services offering with neuro-sciences specific logic following the SOA approach and respecting its cornerstones. They have engaged in the development of an upperware stack of facilities ranging from generic middleware related services to domain specific interfaces closer to end-users. The latter materializes under the form of a thin layer of software services sitting on top of the grid middleware that wraps up and abstracts from underlying technologies to deliver adapted functionality to end-users, while respecting the SOA model.

The following section briefly discusses the design and corse-grained description of this thin layer with a special emphasis on its main pillar, i.e. the workflow management components, and in what extent it conforms to the previously introduced rules for delivering a reusable platform.

This rather incomplete system architecture description aims to mainly give clarity to the approach chased in WP10 and its gridification model specification. For more detailed information on the services portfolio, readers are encouraged to read deliverables D5.1 and D6.1, for which appropriate references have been placed in the introductory "Reference Documents" section.

2.3.3. System Architecture

Turning ongoing requirements analyses into solid initial technical foundations, partners have invested a significant effort at sketching a system architecture for structuring subsequent design and developments. The following diagram, i.e. figure 1, thus illustrates the resulting architecture in terms of logical software layers and corresponding functional areas. It introduces the notion of horizontal versus orthogonal layers, where respectively horizontals provide system functionality, whereas orthogonals address non-functional aspects impacting on horizontals.

Starting from the very bottom of the system, i.e. "Backends Middleware", with IT legacy assets to be used in the project such as grid and database infrastructures, various abstraction levels are then introduced. The most important one, so-called "Backends Abstraction", aims to wrap up underlying backends and to allow partners to develop grid/ database agnostic software while still interacting with given technologies and corresponding specificities. Based on this abstraction, further layers are superimposed which deliver more and more specific functions as distance to end-users shrinks. As such, "Domain Logic" aims at grouping so-called "medical generic services" – e.g. medical querying, medical data acquisition and quality control etc – which could be reused in other medical fields, whereas "Business Logic" only focuses on Neuro-Sciences features, e.g. cortical thickness pipeline, segmentation/ normalization algorithms etc, which are then accessed by end-users through a dedicated web portal exposing specialized interfaces.

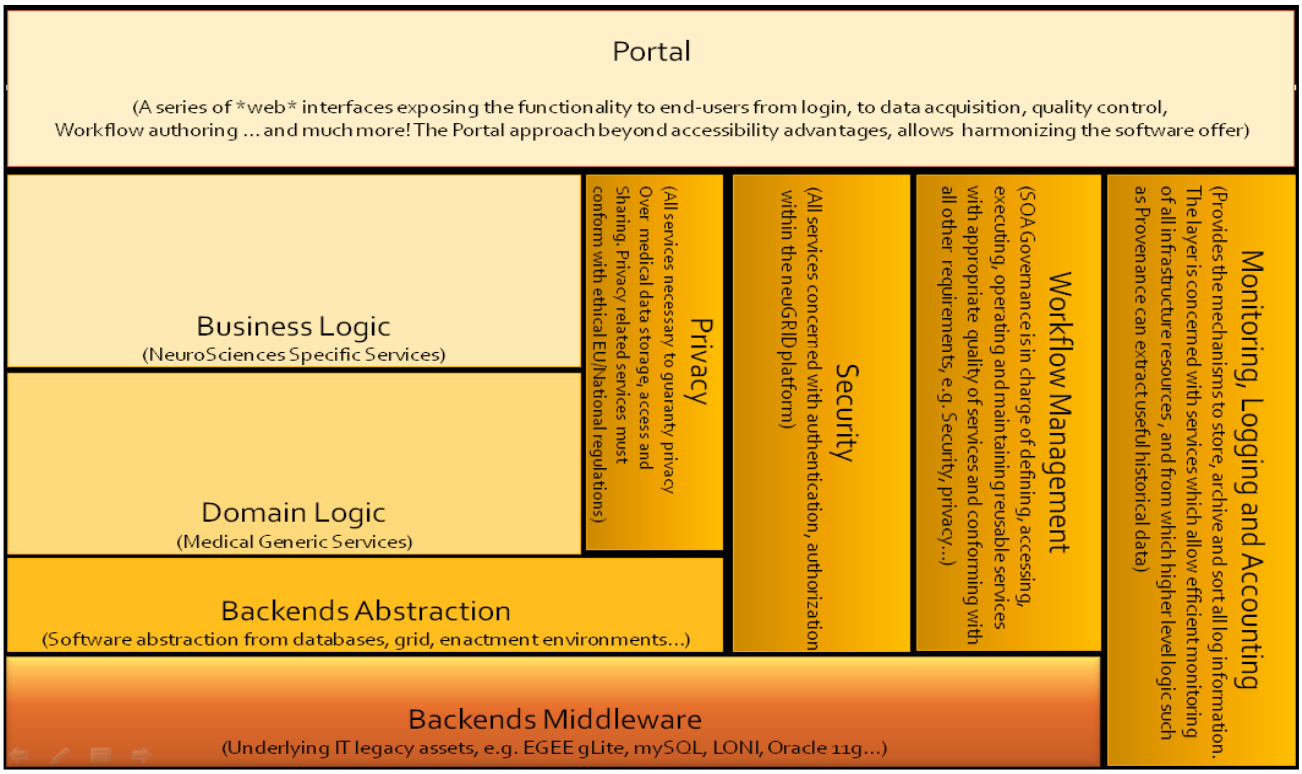


Figure 1. neuGRID System Architecture – Layer View

Orthogonally, the platform aims to offer various means to trace system activitiy via the "Monitoring, Logging and Accounting" subset of services. The neuGRID services are delivered within a secure environment implementing a security scheme as dictated by the requirements, hence having a "Security" layer spanning from the top business logic, to domain logic and finally underlying abstraction. The same applies for privacy aspects since dealing with sensitive data and applications, though essentially impacting on both business and domain logics.

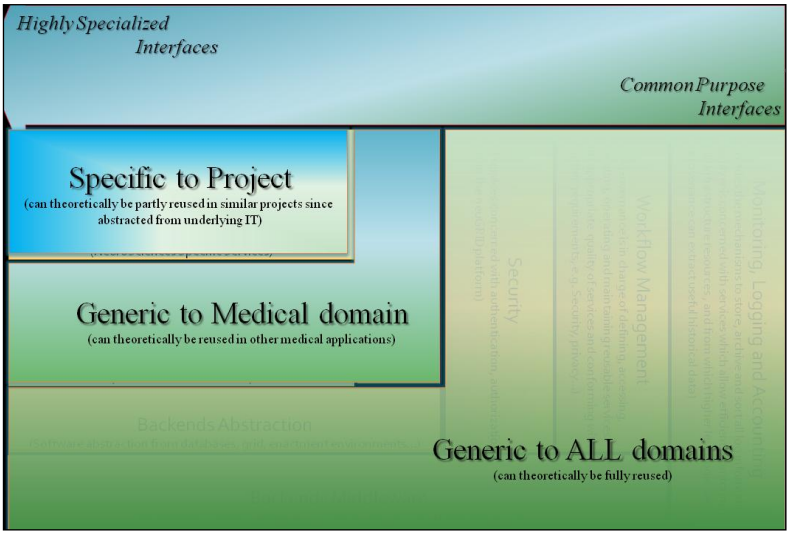


Figure 2. neuGRID System Architecture – Meta-Layer View

This being said, an additional meta-level of layers can be introduced, as illustrated in figure 2 – on the left, where functional and non-functional layers are grouped per levels of reusability. Thus, horizontal layers concerned with backends access/ management, together with orthogonals like monitoring, logging, accounting, workflow management and security can be grouped in a set of artifacts theoretically reusable in all application areas. Similarly, layers such as domain logic and privacy are reusable in other medical fields, while last but not least, the business logic, as its name implies has a much lower reusability spectrum since specialized to Neuro-Sciences.

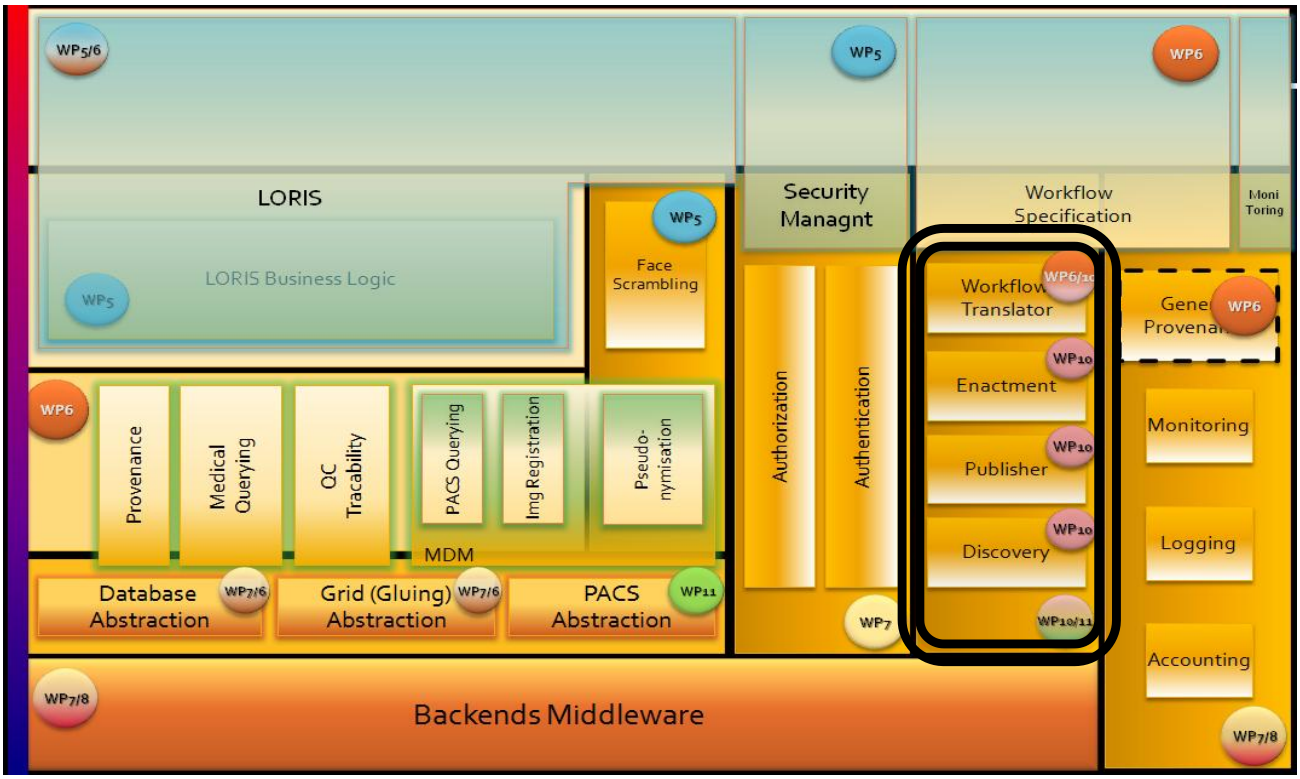


Figure 3. neuGRID System Architecture – Work Packages and Component View

Figure 3 provides more details on the expected services portfolio per layers. Thus one can notice the grouping of functionality in, for instance, (1) the "Monitoring, Logging and Accounting" layer where a dedicated service is introduced per aspect. The same applies to (2) "Security" with authentication and authorization services, (3) "Backends Abstraction" with a functional split between components related to databases, grid and PACS systems access, (4) "Domain Specific" logic with hypothetical medical querying, provenance, quality control, imaging data acquisition and (5) "Privacy" with pseudonymisation and face stripping services.

From WP10 standpoint, key elements of this architecture lie in (6) the "Workflow Management" layer (i.e. surrounded by double black lines in figure 3), where the necessary logic for publishing, discovering and composing functionality is expected to materialize, most likely under the form of service utilities. This othogonal layer intends to supply the key SOA mechanisms – respecting rules introduced in section 2.3.2. Combined with an appropriate gridification model (discussed in next sections), such generic low-level mechanisms will demonstrate the benefits of virtualization when applied to a very focussed area such as Neuro-Sciences, with highly specialized applications.

From their experience in similar projects, i.e. EU-funded FP5 MammoGrid [4] and EU-funded FP6 Health-e-Child [5], the neuGRID technical partners intend to make further progress in the field of medical applications gridification. In particular, there has been significant effort invested and progress made in grid abstraction and web services orchestration within Health-e-Child, which will subsequently be capitalized, tested and compared with related work in the community to address the neuGRID challenges; the idea being to not reinvent the wheel but rather reuse, consolidate and extend solid background.

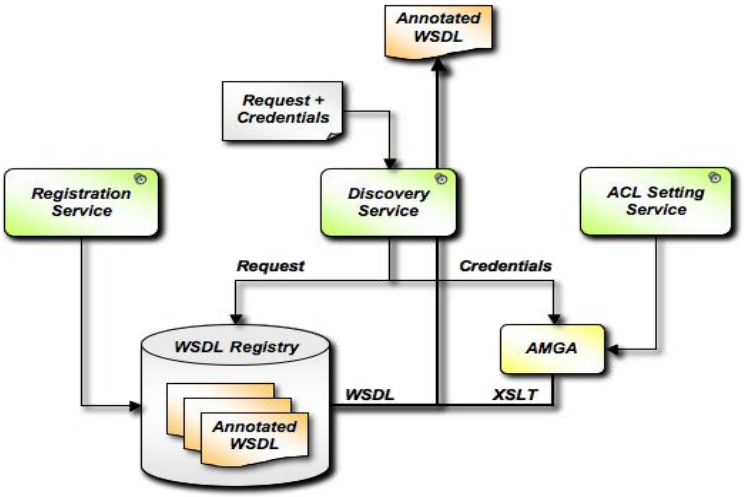


Figure 4. SOA Framework

In particular, the Publication, Discovery and Composition Services from the Health-e-Child Gateway [6] provide advanced facilities to manipulate the SOA offering, based on the latest W3C* standards. Processes can be defined using the standard web services orchestration language and then turned into workflows of services mixing grid and web resources, for execution in a distributed environment. The composition service also supports short running as well as long running processes using state-of-the-art approaches and underlying technologies. Figure 4 above illustrates the different components of this solution.

In any case, different possibilities will be considered following the requirements analysis delivery and tested through concrete prototyping such that the resulting system fully satisfies community's needs. The main assumption in the so far designed and here summarized system architecture is that all gridification approaches can be supported from very low-level batch processing of complex task-based job submission to more advanced and state-of-the-art web services composition, thus opening the pathway to a wide variety of possibilities.

The present document remaining at the design specification level, further technical insights would not add much to clarity, especially with the requirements analysis still ongoing. The next section therefore anticipates and focusses on preliminary conclusions which can be drawn from end-users expectations/ needs and potential tangible gridification model to be applied to Neuro-Sciences toolkits, with a special emphasis on neuro-imaging.

3. Preliminary Pipeline Requirements Analysis

3.1. Complexity in Neuro-Imaging

In the study of neuro-degenerative pathologies and more particularly applied to Alzheimer's disease, various parameters are extracted from imaging that can quantify/ qualify the disease progression/ diagnosis. Parameters such as brain volume change over time, regional changes or even white matter lesions can be extracted by applying different image processing techniques onto patients' brain scans. However, in almost all cases, such extractions cannot be fully automated and require the intervention of an expert to slightly tune the process and/ or clean data.

Let's take as an example the measurement of brain atrophy rates over time, i.e. the amount of brain tissue that is lost by Alzheimer's patients over, say, one year. This measure is relevant in that it is the most valid marker of disease activity available to date and is ideal to test the effect of drugs aimed to slow or arrest its progression.

The first step to undertake in its measurement relates to noise reduction and is aimed to reduce random variations in images due to magnetic field changes and scanner calibration. Here, the MRIcro [7] imaging toolkit is used to correct images manually by checking the homogeneity of the signal over the whole brain. This process cannot be automated and requires trained users in that inhomogeneities and other artifacts may not always be obvious to a lay eye (e.g. blood vessel may look similar to brain tissues, noise can appear around the eyes area, etc).

The second step involves the digital extraction of the brain through segmentation of brain from non-brain voxels*. Here, one of the tools of the fMRIB Software Library (FSL) [8] is used, namely the Brain Extraction Tool (BET). The operator manually selects areas to be included (i.e. according to shades of gray, thresholding, etc) and others that should be omitted from the calculation. The obtained brain volume can be compared to a set of reference brains for diagnostic purposes, or can be registered (i.e. aligned in the 3D space) to a follow-up image to compute atrophy rate.

The latter is calculated using the SIENA [9] software. This gives as output the difference of the brain contours between the baseline and the follow-up image in order to compute the actual shrinkage or increase of the brain size in quantitative terms (in cc or ml), giving a volume ratio directly indicative of the disease progression.

This simple example of a given process that clinical researchers usually go through to extract meaningful imaging markers is highly indicative of the toolkits heterogeneity, the somewhat interactive nature of neuro-imaging pipelines and the complexity inherent to (intermediary) data cleaning and imaging algorithms parameterization. Offering a harmonized environment to run such pipelines therefore suggests a flexible yet powerful gridification* model, which gives enough freedom to researchers to tune processes and interact with the system as needed.

Although it is clearly not the intention of neuGRID to fully develop such a virtualized environment in its current phase, specifying a gridification* model that could facilitate the development of such a useful tool on the long run remains WP10's main objective.

To do so, it has therefore been necessary to undergo a short study of imaging and data mining toolkits being used by clinical researchers within neuGRID. The following of this section therefore attemps to list mostly utilized applications at the three end-user institutions of the project, with subsequent classification according to diverse criteria (e.g. toolkit, software dependencies, imaging features, etc). These classifications then help scoping the nature of such pipelines and algorithms while supporting the formalisation of corresponding use-cases.

3.2. Pipeline Toolkits

The following table lists the pipeline tools that are in frequent use by the research centers as expressed by end-users. It aims to give a taste on the faced difficulty and heterogeneity of available imaging/ mining toolkits, whether commercial suites or community software:

Institute	Pipeline Tools	Analysis Tools
VUmc	fMRIB Software Library (FSL): Flirt, Fnirt, FDT, FAST, Melodic (visualization tool), Siena, XSiena, FEAT, http://www.fmrib.ox.ac.uk • MRIcro, Brain Extraction Tool (BET), http://www.sph.sc.edu/comd/rorden/mricro.html • Montreal Neurological Institute (MNI) (BIC Tools & Software – The Brain Imaging Software Toolbox): N3. http://www.bic.mni.mcgill.ca/software/ • BioInformatics Research Network (BIRN) (Gradiant Non-Linearity Distortion Correction): Gradient non-linearity. http://www.nbirn.net/ • DRG Fluid. • Generic: ○ Image calculations (adding subtracting, multiplying etc) ○ Morphological operations on images ○ File format conversions	Statistical Parametric Mapping – SPM http://www.fil.ion.ucl.ac.uk/spm/software/
KI	• MNI BIC Tool – CIVET Pipeline http://wiki.bic.mni.mcgill.ca/index.php/CIVET , • FSL, • Brainvoyager http://www.brainvoyager.com/ • Matlab http://www.matlab.com , • Analysis fo Functional NeuroImages (AFNI), http://afni.nimh.nih.gov/afni/ • E-prime http://www.pstnet.com/ and • Statistica.	Hermes (Hermes Medical) B-MAP (Pipeline 1 and Pipeline 2) http://www.hermesmedical.com/
FBF	• FSL Tools fMRIB's Diffusion Toolbox FDT 2.0, Melodic • MNI BIC Tools: • Display, register, Brainsuite • LoNI http://www.loni.ucla.edu/Software/ tools: • Dual_warpe_warpcurve, Decoder_blend_all,	• SPSS http://www.spss.com/, • Statistical Parametric Mapping – SPM, Matlab, Quanta 6.1

	mk_seg16bit, mk_gray, add_gray_to_inflated_LEFT1, add_gray_to_inflated_RIGHT1, pmap_apeVSctrl, make_UVL_*; 1st_script_tracer_avg_DIAG; 2nd_script_core_test_L_DIAG; 2nd_script_core_test_R_DIAG; Pmap_DistCore_DIAG
	• MRIcro (MRIcro) (visualization)
	  • BET Function
	• IdeALab Tools (IdeALab) http://neuroscience.ucdavis.edu/idealab/software/index.php
	• Image Conversion software
	  • MRIconverter
	  • dcm2nii
	• New Promising Tools:
	  • 3D Slicer, VTK, Freesurfer, MPIAV, NA-MIC Kit components, MED-INRIA, BrainVoyager, BrainMAP
		• R (R) http://www.r-project.org
		• Statistical Parametric Mapping – SPM

This list demonstrates that end-users develop preferences over time from personal experience and projects, which lead them to use various combinations of toolkits/ algorithms to extract complex features. From these only three centers short survey, one can notice that there are however a few common tools, as highlighted in the following table.

	FSL	MNI/BIC	LoNI	SPM	MRIcro/BET	SPSS	HERMES	Idealab	Matlab	R	AFNI	E-Prime	Statistica	DRG	BIRN	BrainVoy.	QUANTA
VUmc	X	X		X	X									X	X		
KI	X	X					X		X		X	X	X			X	
FBF	X	X	X	X	X	X		X	X	X							X

As a conclusion to this initial comparative table, FSL, MNI/BIC, SPM, MRIcro and Matlab seem to be the most common set of (neuro) imaging and data mining toolkits being used by our Neuro-Scientists.

3.2.1. Pipelines vs Imaging Capabilities

The following table gives a list of popular toolkits and corresponding image processing capabilities used to respectively normalize data, convert image files, anonymize data, extract features from within images and process statistics. This classification aims to introduce the notion of categories, that resulting neuGRID system could use to sort out year 2 gridified algorithms portfolio (see D10.2 at Month 22).

Main Category	Type of Processing	Pipeline / Algorithm	Toolkit
Pre & Intermediar	Normalization	Linear and nonlinear (correction factors)	SPM
		Segmentation (voxels labelling priors-based)	SPM

y Processing		Warping (sulci based) Warping (intensity based)	LoNI MNI
	File Conversion	Dicom to MINC Dicom to Analyze	MNI MNI MRIcro
	Anonymization	Face Scrambling Pseudonymization	LoNI --
Research	Segmentation	Cortical Density Cortical Thickness Hippocampus Atrophy (shrinkage) Hippocampus Volume White Matter Volume and Distribution Cortical Thickness	SPM LoNI LoNI LoNI MNI LoNI IdeALab MNI
	Statistics	Cross Population Patterns	--
Diagnostic	Segmentation	Cortical Density Cortical Contour Drawing + Voxels Counting White Matter Age Related Scale (Wahlund) Regional Brain Metabolism Alterations	SPM MNI -- HERMES

While this categorization will certainly gain in clarity and structuring when formalized as part of the requirements analysis, it is already clear that algorithms/ pipelines can be classified whether they are used to (1) convert, (2) normalize, (3) anonymize data or to (4) extract meaningful measurements through imaging segmentation and (5) process statistical analyses.

Beyond classification and along the lines of thus far gathered requirements, this wide variety of toolkit utilities indicates that there is a generalisable pipeline model. Clearly, four steps tend to shape, as illustrated in figure 5 below (and as is verifiable in end-users' process descriptions – see Appendix A – SPM Pipeline process for a concrete example).

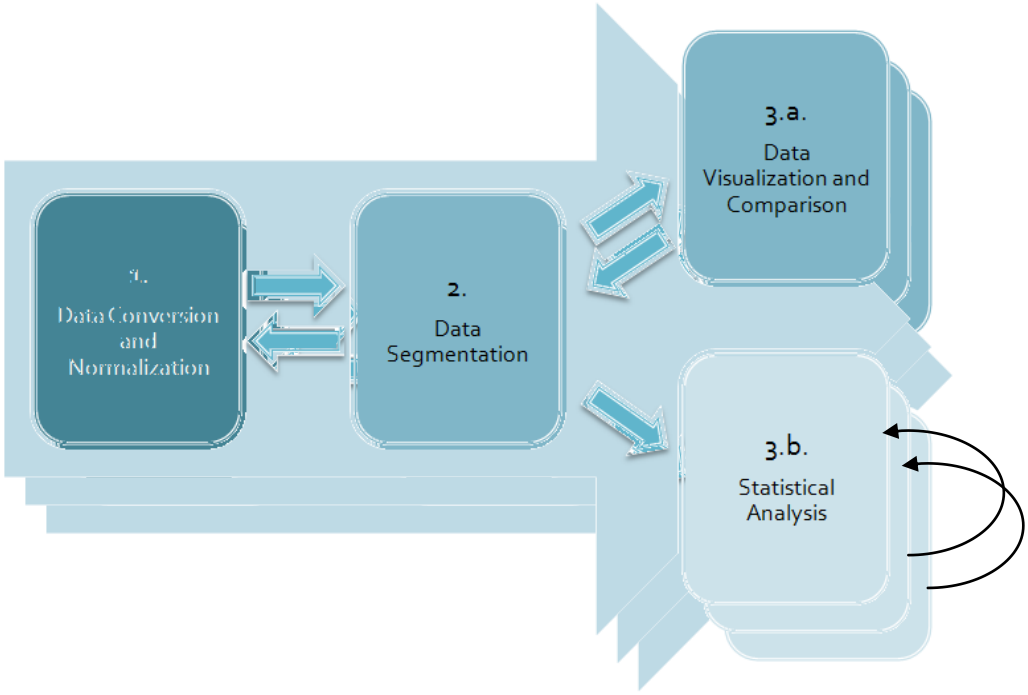


Figure 5. Pipeline Meta-Process

(1) Data Conversion and Normalization
Before executing any pipeline of algorithms, the initial step consists in normalizing the data (i.e. making it comparable) and converting it into an appropriate format for further analysis. In some cases, clinical researchers apply a selection of normalization algorithms and then manually trace brain structures or clean images slice by slice to allow for deeper computer aided analysis. Such manual annotation/ quality control processes are time consuming. For instance, when an expert wants to trace brain structures, it takes approximately:

- For total brain volume: ¼ hour for 1 patient,
- For Hippocampus volume only: ½ hour for 1 patient.

Normalization algorithms are selected according to the modality and quality of data, which can vary slightly from one imaging device to another due to calibration differences. For this reason, normalization from time to time does not work or outputs wrong results. Neuro-Scientists therefore have to assess the quality of the output data, thus introducing a human interaction requirement in the loop. Normalization is a "no-return" process; this is the reason why original data is always kept in a separate place. All processing steps are usually traced and intermediary data also stored in a separate folder, for the very same reason.
Note: the data acquisition process is not taken into consideration in the present.

(2) Data Segmentation
Once the data has been quality assessed, a concrete measurement is extracted. In the context of neuGRID, researchers usually investigate morphological or functional changes and lesions by measuring for instance how thick the cortex is, from structural imaging. This is done by applying a given pipeline of image processing algorithms onto brain images. Such pipelines are either existing/ tested ones that a researcher applies straight away onto his dataset or a pipeline freshly specified from the combination of different algorithms and sometimes fragmented between different toolkits.

Once a given pipeline is executed, researchers in almost all cases have to check intermediary data quality, i.e. data produced at the various stages of the pipeline. This assessment is again operated visually and can lead to additional data cleaining or even re-execution of the concerned pipeline step(s), so that subsequent processing is successful. Recalling figure 5, there is therefore a cycle established between steps (1), (2) and (3) while a given pipeline is running. Also noticable from discussions with end-users, the expertize related to pipelines (i.e. algorithms parameters, wokflow/ pipeline description, etc) is stored in a separate report using an ad-hoc format. In other words, the knowledge associated to a given pipeine is never expressed using a standard notation (nor turned in machine-processable specifications).

(3) Data Visualization and Comparison
As formerly introduced, data visualization can occur at various stages of the pipeline. It can be operated at the outset to visualize the resulting extraction or after given steps of the pipeline execution in order to visually check the output quality of concerned algorithms. In the latter case, visualization supports the quality control process, whereas in the former it allows end-user to validate his measurement or pipeline, as well as to compare the measurement with other experiment results or references.

(4) Statistical Analysis
Recalling the section introductory example, statistical analysis may be applied for interesting measurements onto a large set of patients' scans. In the case of brain atrophy for instance, it would mean running the MRIcro and FSL pipelines several times, as is illustrated in figure 5 with a series of arrows on the right, onto different patients' brains and corresponding follow-ups to obtain an indicative atrophy percentage over a given population.

3.2.2. Pipelines vs Software Characteristics

This last table provides detailed information about popular pipeline toolkits in terms of supported Operating System(s) (OS), licensing conditions, programming languages and data formats, while recalling available imaging features. This is useful to understand the potential difficulty which will be faced to gridify a given toolkit.

Criteria / Pipelines	OS	Licensing	Prog. Language	Supported Data Format	Features
SPM	OS Independent	GPL*	Matlab	Analyze, NIftTI-1*, MINC*	Images Visualization Segmentation (apriori-based) Registration (linear and affine) Warping (Jacobian) Volumetric Analysis (density and volume) fMRI analysis PET analyses
FSL	MacOS X, Windows NT / 2000, Linux and SunOS / Solaris	FSL* License	C / C++	Analyze*, NIftTI-1*	Images Visualization (FSLview) Segmentation (FAST) Registration (FLIRT) Affine Warp cross-sectional (SIENAX) and longitudinal (SIENA) Volumetric Analysis fMRI analysis (FEAT) Independent Component Analysis (MELODIC) Tractography (FDT) Diffusion tensor voxelwise analysis (TBSS)
LoNI	MacOS X, Windows NT / 2000, Linux and SunOS/Solaris	LoNI Software Licence	Java	AFNI BRIK, Analyze*, bshort / bfloat, DICOM*, MGH/MGZ, MINC*, MINC2, NIftTI-1*	Image conversion (MNI toolkit) Non-uniformity correction (MNI) Segmentation (MNI) Warping (sulci based; flat maps) Image visualization (DISPLAY)
IDeALab	Linux (Fedora core) + SunOS/Solaris	PV-Wave & Quanta license	PV-wave, Shell Scripting	Analyze*, Quanta, Interfile	Image conversion Images Visualization (sv) WMHs Segmentation (Quanta) Linear Registration Warping (Spline)
B-MAP / HERMES	Unix for backend, Windows for frontend	HERMES Commercial Licence	---	---	Image Conversion, Interpolation, Template of reference brains, Masking of extra-cranial tissue, Morphing, Signal Inhomogeneity (Bias field), Tissue Segmentation (Gaussian

					Estimation, Fuzzy Cluster Analysis), Segmentation with ROI Analysis.
MNI	MacOS X, Linux and SGI IRIX	---	C, Perl, (some Java for visualization)	MINC*	Image Conversion, Images Visualization, Anatomical Regions Labelling, Sulcal Extraction, Cortex Extraction, Image Resampling, Statistical Analysis

This table gives concrete technical hints on the toolkits gridification* applicability. Indeed, it shows that most of them are not cross-platform, except SPM. Toolkits accept potentially different file formats – although image converters exist – and last but not least, toolkits are developed under diverse programming languages.

From interaction with end-users and with concrete demonstrations during requirements meetings, it is noticeable though that in spite of these differences, (almost) all toolkits algorithms materialize under the form of Unix-like binaries/ scripts/ libraries. This simplifies greatly, if not eliminates – technically speaking, the problems related to data flows in such pipelines. Indeed, by doing so algorithms only have to deal with simple input and output types such as strings of characters, whether being a configuration value for the algorithm itself or a physical path to target image files. This strengthens and confirms the grid relevance and applicability to neuGRID.

The next section presents an extract of ongoing requirements analysis with an end-to-end use-case illustrating the spectrum of functionality that neuGRID is attempting to satisfy. The section then delivers preliminary conclusions by qualifying pipelines and thus introducing possible gridification approaches.

3.3. End-to-end Use-Case

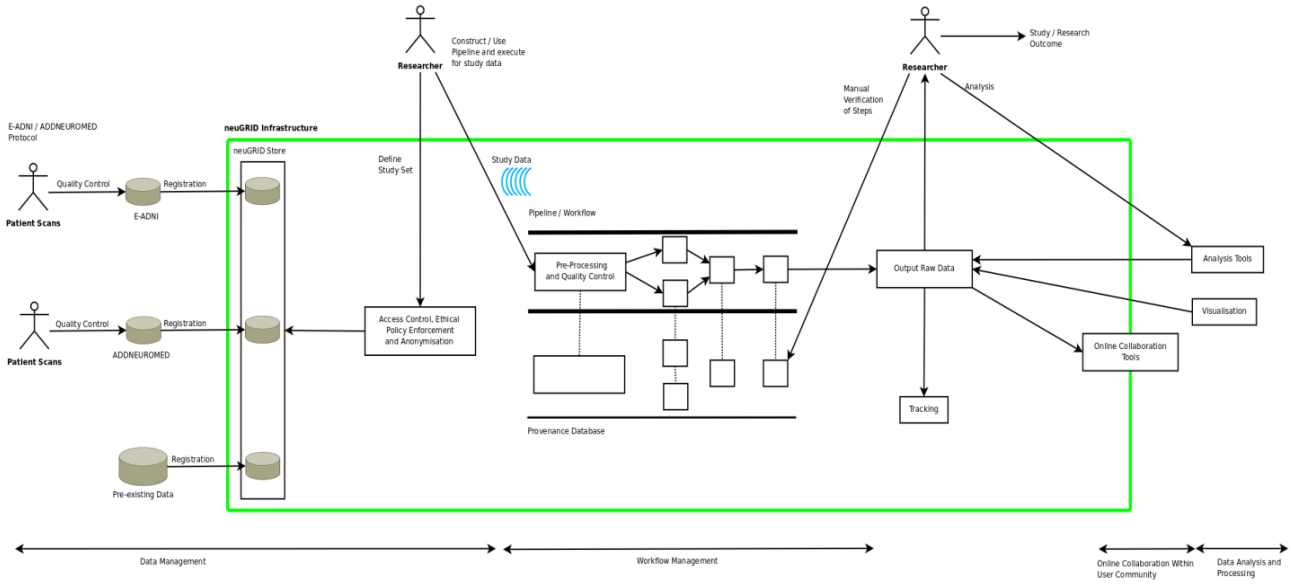


Figure 6. End-to-end Use-Case

Figure 6 above introduces neuGRID's end-to-end use-case, which attempts to give an overview of the system functionality from initial data acquisition on the left, to validation of scientific workflows and their sharing on the right, as well as provenance data analysis at the bottom.

As far as workflows/ pipelines requirements are concerned in the middle of figure 6, a number of indicative use-cases have been formulated thus far, which have undergone detailed specification for delivery in D9.1 at Month 14, such as:

- Construct, visualize, annotate and edit new workflows,
- Work with draft workflows and use version control to manage them,
- Visualize, annotate and edit existing workflows,
- Search for existing research sets or define new groups of images and other information to be processed using the workflow,
- Run, monitor and control the execution of a workflow. This would involve perhaps the ability to cancel, edit and restart an execution
- Search for and select the desired analysis pipeline from a set of existing workflows, edit settings if required and execute
- Search the history of a given workflow to find a particular version of it for use in a specific piece of research
- Store a history of each workflow execution, research set and settings. Allow user annotation of such information
- etc

Indicative use-cases have been sorted out per functional areas like "Workflow Specification and Development", "Workflow Execution and Management", "Validation of Workflows", "Sharing of Workflows" and corresponding requirements are being formalized and prioritized. The following table attempts to however list draft requirements as were gathered from end-users.

(A) Workflow Specification	
1	*The system should offer ways to store/ provide information about data that a given workflow cannot process. Some sorts of annotations and corresponding interfaces would be helpful. These annotations, at a first glance, could reside at the level of data rather than at the level of the workflow description, as raised by the users. Such a feature would prevent from executing workflows on inappropriate data.*
2	*Workflows and algorithms should be categorized and documented. Newly designed workflows and/or existing algorithms should be stored and sorted out according to some project/standard classification. Some of the popular neuroimaging toolkits do have their own classification/terminology, which could be reused (and possibly extended with additional features).*
3	*Workflows should be versioned and their descriptions contain information about underlying toolkits versions as well. That would allow the platform to keep up with the speed of toolkits evolutions, while ensuring that former workflows can still run with appropriate versions of concerned libraries.*
4	*Eventually the system should offer the possibility to edit the source code of given algorithms.*
5	*Workflow descriptions should be stored, made editable and reusable in other workflows. This authoring environment should allow users to easily navigate through their previous descriptions, import, extract, share, and execute on demand.*
6	*Eventually the system should offer possibilities to share workflow descriptions with other researchers. As of today, such sharing materializes under the form of scripts emailing and accompanying technical explanations to run those.*
(B) Workflow Validation	
1	*The system should offer instrumentation tuning possibilities. Workflow descriptions should be accompanied with test data samples that can be run at any time for validation purposes (i.e. a "make test").*
2	*The system could introduce the notion of workflow execution "goodness" to allow comparison of workflow parameters tweaking and to verify workflows coherence on the long run.*
3	*The system could eventually provide batch test processing and statistics about workflows executions, which would allow monitoring published workflows continuously.*
4	*The system could eventually propose appropriate viewing applications when a user would like to*

	verify a given workflow/processing step.
(C) Workflow Execution	
1	*The system should allow re-processing inner steps of a workflow being executed, at the request of a user. For instance, in case the output of a certain task/execution step is not satisfactory, user should be able to modify some of its execution parameters and/or reprocess it on demand.*
2	*Workflows are made of processing steps and sometimes integrate other predefined workflows. Workflows usually have pre-processing/processing/post-processing phases.*
3	*Workflows are essentially sequential and asynchronous in nature (very similar to shell-like applications). Very rarely such workflows can be designed distributed as they act on a well-defined and single location dataset. Sometimes users need to execute several workflows in parallel, meaning groups of workflows being executed in different locations, while the workflow itself is run on a single computer.*
4	*The system should allow holding, resuming and stopping workflows being executed at any moment at the request of the user.*
5	*The system should keep track of workflow execution logs from data access, to data modification and workflow execution errors of any type.*
6	*Workflow execution should be interactive with possibly long and short runtimes. A completed workflow or any of its inner steps should be verifiable by the user. At any moment the user should be able to check the output of a given step by visualizing or reading the outcome with appropriate applications.*
7	*The system should store the output of workflows, which can be of different types, ranging from ad-hoc files, to structured XML resultsets or images.*
8	*The system should guaranty toolkits/ libraries versions before executing a given workflow.*
9	*The system should allow designing workflows of algorithms coming from different toolkits. That would allow mixing different toolkits and would open the pathway to new interesting combinations.*
10	*Inner steps of workflows should be considered as reusable building blocks (i.e. black/grey boxes). Eventually the system could offer generic wrappers to attach new algorithms/ workflows or tune existing ones. This functionality is actually important as users usually have to adapt existing algorithms' output to fit the input of subsequent ones.*
11	*The system should take care of automatically converting the data to appropriate formats when executing inner processing steps of workflows. It therefore should know about data format requirements for all of the available algorithms/workflows in the system.*
12	*The system should allow when possible to batch process data and offer scheduling optimization means accordingly in the case of automated pipelines.*
(D) Workflow Authoring Interface	
1	*The system should allow workflow authoring in different ways. The corresponding interface(s) should therefore allow from simple graphical design of workflows, to more advanced "live" scripting. Scripting is the favored interface, i.e. users tned to not trust graphical user interfaces (GUI) much and fear to be limited (e.g. GUI limitations of MIDAS toolkit).*
2	*The workflow GUI should be independent somehow of the underlying platform actually executing the processes. A user should have the possibility to use one or the other independently.*
3	*The interface should offer/ integrate visualization tools to allow for data verification, result browsing and so-forth. Ideally the system should be able to integrate with existing visualization tools, that users are used to use.*
4	*The interface should offer workflow execution logs browsing facilities, to allow user to efficiently trace their executions and identify bad performing processing steps.*

Note: design constraints are still being analyzed and are thus not presented in this document.

3.4. Requirements Analysis Conclusions

3.4.1. Generalities

From the so far formalized requirements and studied pipelines toolkits, a number of conclusions can be drawn. It has been consensually agreed that an analysis pipeline* corresponds to so-called workflow* in computing terminology. In the context of the project a workflow might be defined as a structured process and data flow through a series of steps that can be executed by a user.
Analysis techniques may take an image as an input and return a new image with certain features enhanced, other methods can result in statistical output. This illustrates that there is a huge range of variations in the complexity of workflows that are required. Each research project/ study is likely to have a different set of requirements. It is clear therefore, even at this early stage of the project that a highly flexible and adaptable workflow construction and execution environment is necessary.

The construction of new pipelines or workflows is a difficult task. Results need interpretation at each stage in the process if errors and bad results are to be avoided. An understanding of the underlying algorithms is required to be able to do this. Many algorithms are for generic image processing and are not specially designed for clinical purposes. This means that there can be some difficulties in bridging the gap between theoretical image processing and clinical requirements. Experience is vital to the successful handling of this process. Initial processing of the images is also often required to ensure that they all conform to a standard format that can be accurately compared. In some cases the final results need further interpretation if meaningful conclusions are to be reached.

A major issue is that image processing and analysis can take a long time to complete. Pipelines commonly run batches of each step on each image, steps that paradoxically are in most cases short runtimes. This means that for of a study of say 500 people, the pipeline would carry out step one on each image or 500 times, before taking the output and going on to step 2 which would be carried out 500 times and this would continue until the end of the pipeline was reached.

Fault tolerance is also important because a failure at any point during the execution of a pipeline may invalidate the results that are produced. Images must therefore be selected carefully otherwise inconsistencies in them may have unintended consequences. There are examples where studies have proven that erroneous results can creep in when care is not taken during image selection. Bad images are normally discarded early on in the process. Some analysis steps take a few seconds others several hours, especially if large studies are involved. A robust means of running, execution tracking and speeding up complex processing algorithms would be useful to researchers.

3.4.2. Pipelines' Nature

By extracting from the design objectives formulated in section 3.3 only those related to pipelines, which are considered to potential impact on the gridification model, one may generalize intrinsic pipeline characteristics as follows:

1. Pipelines encompass ***Significant Added-Value***: pipelines are not just sequences of algorithms. They encompass domain knowledge which is essential to Neuro-Scientists. Their descriptions thus have to incorporate such knowledge	A.1, A.2, A.3, A.6, D.1, D.2
2. Pipelines are ***Heterogeneous***: pipelines utilize various technologies/ environments and sometimes are fragmented across different toolkits	C.9, C.10
3. Pipelines are ***Interactive***: outputs of inner steps have to be checked in most	C.1, C.6

cases to assure successful execution of following ones	
4. Pipelines are ***Iterative and Recursive***: in case of bad outputs, pipelines or inner steps have to be executed again. Pipelines can also be composite, i.e. pipeline of pipelines	C.2, C.4
5. Pipelines are mainly ***Task-based***: processing steps are in most cases executable code enacted using ad-hoc or scripting languages describing command lines and associated parameters	C.3, C.10
6. Pipelines are mainly ***Sequential***: they in most cases consist of a series of steps executed in series (especially true for voxel*-based image processing). Few cases require parallelism (or could be parallelized), taken aside statistical analyses where the same pipeline is run onto a large dataset thus executable in batches over multiple processing nodes to optimize overall runtime	C.3
7. Pipelines are ***Computing Intensive***: image processing algorithms used in pipelines usually have short runtimes but are applied to several images and many times, thus making overall pipelines processing times quite long	C.12
8. Pipelines are ***Data Intensive***: image processing algorithms usually output intermediary imaging data for inputting in next steps of the pipeline. Pipelines thus tend to produce 'n' times the initial dataset volume, where 'n' is most likely equal to the number of segmentation steps	C.7

3.4.3. Pipelines' Anatomy

From formerly described nature, current pipelines in Neuro-Sciences constitute a very good case for gridification. One common characteristic seems to clearly shape, which is the form under which inner algorithms materialize, whatever toolkit they are part of. Indeed, imaging algorithms are mainly about Unix-like binaries/ scripts/ command line interfaces (CLI) accepting/ producing simple strings of characters respectively as input parameters and/ or as output values. This is what figure 7 illustrates below (by recalling the conceptualization introduced in section 3.2.1, i.e. figure 5).

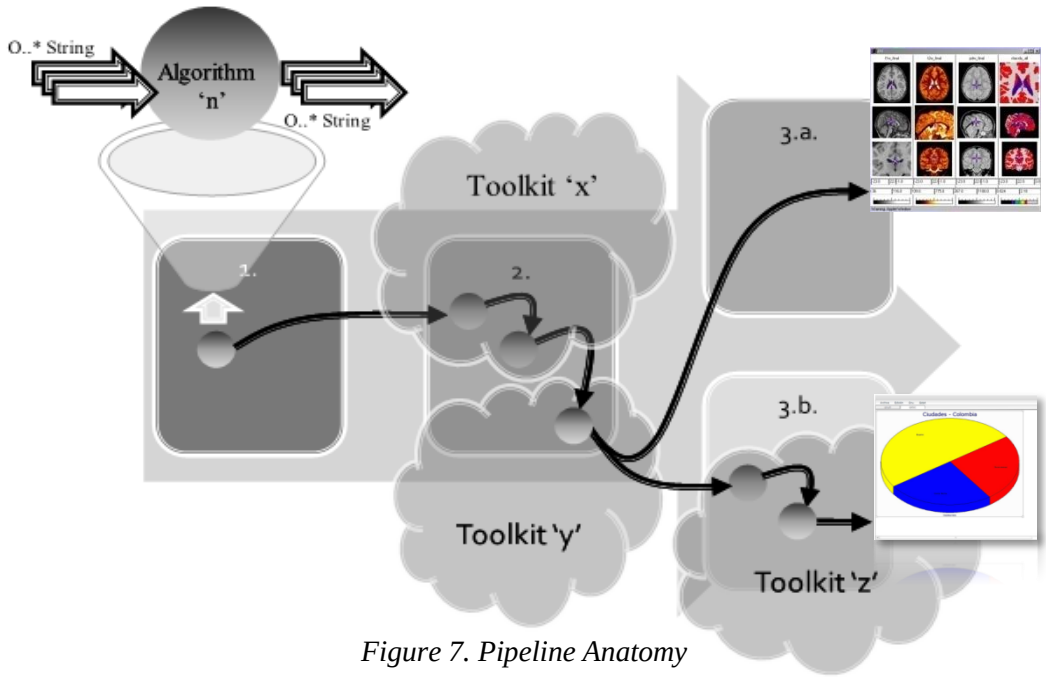


Figure 7. Pipeline Anatomy

In figure 7, a complex pipeline is illustrated which combines algorithms from three different toolkits, but where algorithms themselves have the same anatomy. For a more concrete example of pipelines' anatomy, see "Appendix B – CIVET Pipeline".

The next section introduces the gridification approach and resulting model that authors want to bring forward. An initial high-level description of the model is provided to demonstrate in what extent it addresses pipeline specifities but also how it could cope with future extensions.

4. Design Specifications

4.1. Gridification Introduction

Various projects around the globe are utilizing grid infrastructures to support biomedical applications. In the US, most notably the caBIG™ initiative (http://cabig.cancer.gov/) founded by the National Cancer Institute (NCI) in 2004 to speed discoveries in cancer research by linking research institutes and healthcare providers by enabling "*the collection, analysis, and sharing of data and knowledge along the entire research pathway from laboratory bench to patient bedside*". In Europe, Information and Communication Technologies (ICT) for health is one key component of the Sixth and Seventh Framework Programmes by the European Commission. Several projects besides neuGRID combine biomedical applications and grid infrastructures, for example:

- @neurIST (http://www.aneurist.org) focuses on integrative decision support systems based on multi-scale computational suites for personalised brain aneurysm rupture risk assessment and treatment,
- ACGT (http://www.eu-acgt.org/) develops semantic and grid-based technologies in support of post genomic clinical trials in cancer research, targeting mainly breast cancer and paediatric nephroblastoma,
- ViroLab (http://www.virolab.org/) develops a virtual laboratory for infectious diseases that facilitates medical knowledge discovery and decision support by allowing, e.g. to relate genotypes to drug-susceptibility phenotypes, and
- Health-e-Child (http://www.health-e-child.org) focuses on clinical decision support and knowledge discovery systems in paediatrics based on vertically integrated data for assisted diagnosis and personalised treatment of cardiomyopathies – Tetralogy of Fallot (ToF), juvenile idiopathic arthritis and brain tumours – gliomas.

Further related research includes applying grid technologies to content based image retrieval in radiology [10] and distributed medical image analysis [11], as well as efforts aiming at semantic data integration allowing services to be composed into meaningful workflows [12].

Thus, significant work has already been pursued in the area of applications migration to the grid. So-called gridification is concerned with porting or developing projects/ applications business logic to software jobs that can further be scheduled in a grid environment. Depending on the application nature and underlying grid technology, this process can become very complex and invasive. While executing non-interactive monolithic Unix-like sequences of CLIs* in grid middleware such as EGEE gLite [13] or Globus [14] can be straight forward (when scheduling optimizations are not considered), it is not the case for modern parallel modular applications involving human interactions and asynchronism. Things get even more complicated when one wants to make full use of the grid capabilities with an application that was not originally designed for running in such distributed environments. In this case, reengineering might be needed; gridification may become highly invasive and last but not least introduce execution overheads.

Accompanying today's major fundamental grid paradigms, there are two conceptual approaches distinguishing which address this challenge. On the one hand, so-called task-based job submission relates processing to executable code described as a computation task, hereinafter referred to as "gridification". On the other hand, so-called service-based execution handles processing as workflows of web services orchestrated in a surrounding SOA, hereinafter referred to as "servitization". While the former applies well to some of the problems faced in neuGRID (i.e. see pipelines' nature, points #2, 5, 6, 7 and 8) and could constitute an interesting short term solution, it does not abstract end-users from the grid specificities nor does it facilitate interactivity, and depending on applied scheduling policy, can introduce considerable overheads (e.g. when executing multiple short runtime stages such as the ones most likely to be faced in neuro-imaging).

While gridification and servitization greatly differ in principles – e.g. data input/output, discovery mechanisms, etc – the present design specification argues interesting complementarities, to address neuGRID's objectives. This is what the next section elaborates on.

4.2. Gridification Approach and Model

The approach chased in WP10 is one that advocates a hybrid model sitting in between gridification and servitization of the business logic. Indeed, it is authors' belief that using jointly both concepts would significantly help addressing all formerly raised specificities of neuro-scientific pipelines and introduces the necessary flexibility to accommodate with new applications integration on the long run, especially thanks to the virtualization/ abstraction dimension brought in by the SOA paradigm.

In particular the model here presented is based on former investigations conducted in the Health-e-Child project [15] and by collaborators involved in the development of the so-called MOTEUR workflow engine [16]. The former pionneered a sound SOA framework to efficiently and rapidly create secure simple, ubiquitous, loosely-coupled and stateless web services (see section 2.3.2 for detailed explanations of SOA rules). The latter introduced the notion of a generic web service wrapper [17] embedding legacy codes in service-based workflows (see [17] for an exhaustive review of legacy code wrapping approaches).

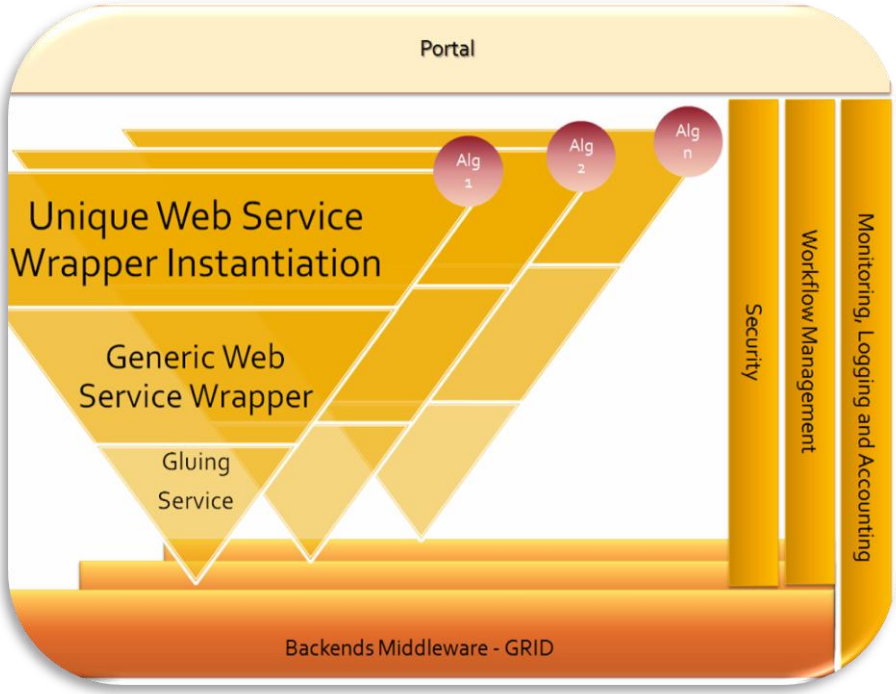


Figure 8. Gridification Model

Authors' aim is therefore to integrate, extend and complete the above concepts based on respective concrete investigations and scientific conclusions.

The survey of neuro-sciences toolkits (and in particular [19]) conducted for the sake of requirements analysis, has shed light on interesting practices in the community, which when contrasted with current Web services workflow authoring environments has opened the pathway to hybrid thinking.

This is what figure 8 above illustrates, by recalling the ad-hoc representation used in section 2.3.3. The proposed approach relies on the following 3 key points:

- (1) Using a so-called generic "gluing service" to submit job execution orders to underlying grids (see deliverable D6.1 for more information on neuGRID's gluing service design specifications). The gluing service abstracts upper layers of the system from grid specificities and is responsible for actual job submission. Note that this is in line with the conclusions of [18].

- (2) Using a generic web service wrapper to embed legacy code and optimize resulting job/ pipeline scheduling at execution time, which is of absolute relevance in the context of neuro-imaging toolkits, given their algorithms non-functional similarities.

- (3) Instantiating a unique web service wrapper per algorithm/ pipeline to be published in the neuGRID SOA, thus allowing (both atomic and composite) processing tasks to be discovered, composed and subsequently published as new ones. See [19] for a similar approach with different implementation and technology.

Conceptually speaking each of these 3 substrates, plays a different but key role. While (1) introduces abstraction from grids and thus allows interacting with a variety of middleware, (2) takes care of appropriately parameterizing (1), characterizes commonalities of algorithms/ pipelines and opens a broad avenue to job scheduling optimization techniques (e.g. jobs grouping). (3) on the other hand and beyond parameterizing (2), turns this ecosystem of virtualized neuro-utilities into a set of publishable, discoverable and composable entities, which are very close to end-users' expertize.
Note that (3) slightly differs from the approach undertaken in [18], as it is a direct consequence to the strict application of SOA's rules, as presented in section 2.3.2. The expected result is a service that can be used directly by end-users to execute a given algoritm.

The combined use of these three elements within neuGRID's SOA is believed to constitute a tangible solution to address formerly gathered pipelines' characteristics. See the following table for a detailed mapping (recalling table introduced in section 3.4.2 and focussing on aspects which are not obvious ta tackle in a task-based approach):

1. ***Added-Value***	Pipelines can be specified as workflows of web services, in spite of having concrete algorithms published in the grid. Current W3C standards allow describing such complex workflows and encompassing semantics/ annotations to store and machine-process associated knowledge/ expertize (see WSBPEL - **http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel**, and SAWSDL - **http://www.w3.org/2002/ws/sawsdl/**)
2. ***Heterogeneity***	Exposing algorithms and pipelines as Web services makes them virtualized/ abstracted and thus allows composing new pipelines with algorithms coming from potentially different toolkits and running in different environments
3. ***Interactiveness***	Web services are naturally indicated for satisfying such requirements as they act as blackboxes triggered by an orchestration entity within the SOA. The orchestration entity and underlying workflow capability are the ones basically offering such interactiveness
4. ***Iterativeness and Recursiveness***	While virtualizing algorithms through web services, the introduction of a generic web service wrapper allows applying scheduling optimization techniques. In the case of highly recursive pipelines of short runtime algorithms, optimization could be obtained by grouping jobs prior to submission to the grid

5. Conclusions and Future Work

This document presented a preliminary analysis of the neuGRID project end-users and system requirements, with the aim of formulating a set of design objectives, constraints and conclusions. By doing so, authors engaged in a survey process, which helped them understanding better the faced issues. Following this, a significant effort of conceptualization and formalization has been invested (and still is ongoing) to produce a relevant analysis as well as a first gridification model to be applied to neuro-scientific pipelines of algorithms.

The proposed model has been designed from past but similar experiences and is based on early requirements analysis conclusions. It constitutes an interesting mix of existing technologies while attempting to bring the SOA benefits closer to end-users. The model is expected to evolve as further prototyping tests will be undergone.

In particular, the intention is to apply it to the Cortical Thickness neuro-imaging pipeline as an interesting validation case study for delivering detailed results at the end of project year 2. Beyond concept validation, this case study will also help improving the corresponding implementation.

Last but not least, the proposed approach is also anticipated to open the pathway to interesting side research work such as the use of model driven engineering (MDE) techniques [20, 21] within the generic Web service wrapper to dynamically change scheduling policies (e.g. grouping optimization vs isolation, middleware selection, etc), as well as within the orchestration entity to address non-functional aspects such as Ethical, Legal and Socio-Economical (ELSE) constraints.

Bibliographical References

[1] Arsanjani, A. (2004). Service-oriented modeling and architecture, IBM technical article. Retrieved October 20, 2008, from http://www.ibm.com/developerworks/library/ws-soa-design1/ .

[2] C. M. MacKenzie et al. Reference Model for Service Oriented Architecture 1.0, OASIS Committee Specification 1, 2 August 2006

[3] Papazoglou, M.P. (2003). Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the 4th International Conference on Web Information Systems Engineering, WISE 2003, Roma, Italy*.

[4] McClatchey, R., Manset, D., & Solomonides, T. (2006). Lessons learned from MammoGrid for integrated biomedical solutions. In *19th IEEE Int. Symposium on Computer-Based Medical Systems, CBMS 2006*.

[5] Health-e-Child, The EU FP6 Information Societies Technology Project. (2008). Retrieved October 20, 2008, from http://www.health-e-child.org/ .

[6] "Health-e-Child: A Grid Platform for European Paediatrics" K. Skaburskas, F. Estrella, J. Shade, D. Manset, J. Revillard, A. Rios, A. Anjum, A. Branson, P. Bloodsworth, T. Hauer, R. McClatchey, D. Rogulin Proceedings of the 2007 Computing in High Energy and Nuclear Physics International Conference - CHEP07, (USA), 2007 Publication in Journal of Physics: Conference Series.

[7] MRIcro. http://www.sph.sc.edu/comd/rorden/mricro.html

[8] FMRIB Software Library. http://www.fmrib.ox.ac.uk

[9] http://people.cs.uchicago.edu/~hai/tmp/doc/siena/index.html

[10] Oliveira, M. C., Cirne, W., & de Azevedo Marques, P. M. (2007). Towards applying content-based image retrieval in the clinical routine. *Future Gener. Comput. Syst.*, 23 (3), 466-474.

[11] Bellotti, R., Cerello, P., Tangaro, S., Bevilacqua, V., Castellano, M., Mastronardi, G., De Carlo, F., Bagnasco, S., Bottigli, U., Cataldo, R., Catanzariti, E., Cheran, S. C., Delogu, P., De Mitri, I., De Nunzio, G., Fantacci, M. E., Fauci, F., Gargano, G., Golosio, B., Indovina, P. L., Lauria, A., Lopez Torres, E., Magro, R., Masala, G. L., Massafra, R., Oliva, P., Preite Martinez, A., Quarta, M., Raso, G., Retico, A., Sitta, M., Stumbo, S., Tata, A., Squarcia, S., Schenone, A., Molinari, E., & Canesi, B. (2007). Distributed medical images analysis on a Grid infrastructure. *Future Gener. Comput. Syst.*, 23 (3), 475-484.

[12] Budura, A., Cudré-Mauroux, P., & Aberer, K. (2007). From bioinformatic web portals to semantically integrated Data Grid networks. *Future Gener. Comput. Syst.*, 23 (3), 485-496.

[13] glite, A lightweight middleware for grid computing. (2008). Retrieved October 20, 2008, from http://glite.web.cern.ch/glite/ .

[14] The Globus Alliance: http://www.globus.org/

[15] "Gridifying Biomedical Applications: Experiences of the Health-e-Child Project" D. Manset, F. Pourraz, A. Tsymbal, J. Revillard, K. Skaburskas, R. McClatchey, A. Anjum, A. Rios, M. Huber Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare. Information Science Reference (IGI Global). Submitted - still being evaluated.

[16] Tristan Glatard, Johan Montagnat, Xavier Pennec, David Emsellem, Diane Lingrand. *"MOTEUR: a data-intensive service-based workflow manager"* Research Report I3S, number I3S/RR-2006-07-FR, 37 pages, Sophia Antipolis, France, mar 2006

[17] Diane Lingrand, Johan Montagnat, Tristan Glatard. *"Estimating the execution context for refining submission strategies on production grids"* (workshop) in Proceedings of the Assessing Models of Networks and Distributed Computing Platforms (ASSESS / ModernBio) (CCgrid'08), pages 753 -- 758

[18] Tristan Glatard, David Emsellem, Johan Montagnat. *"Generic web service wrapper for efficient embedding of legacy codes in service-based workflows"* (workshop) in Proceedings of the Grid-Enabling Legacy Applications and Supporting End Users Workshop (GELA'06), pages 44--53, Paris, France, jun 2006

[19] LoNI Software Tools. http://www.loni.ucla.edu/Software/

[20] "A Formal Model-Driven Approach for Grid Application Architectures" D. Manset, H. Verjus. Technical Report LISTIC No 07/02, University of Savoie - LISTIC, 2007.

[21] "Managing Separation of Concerns in Grid Applications Through Architectural Model Transformations" D. Manset, H. Verjus & R. McClatchey. Lecture Notes in Computer Science Vol 4758 pp 308-310 ISBN ISBN 978-3-540-75131-1 Springer-Verlag, 2007.

Appendix A – SPM Pipeline Process Example

PROCESSAZIONE DELLE IMMAGINI PET (SPM2)

Creo lo stack analyze PET con MRIcro e **flippo con MRIcro** sia le immagini **PET** (xPET) che le immagini **RM** xxx (xxxx), in modo da averle in convenzione neurologica (in modo da non dover modificare il file spm_defaults, che forse coinvolge solo la visualizzazione, nei vari passi della processazione)

Prima della processazione **seleziono la convenzione neurologica nel file "spm_defaults"** (la lascio inalterata x tutta la processazione)

Setto la commissura (con SPM99)
HDR Edit
Options
Set Origin – x,y e z coordinate of voxel
Options
Apply to images (selezionare l'immagine in cui fissare il punto specificato come immagine)
Options
Quit
(controllare di volta in volta che la commissura sia stata fissata in modo corretto comparando il nuovo soggetto al primo con Checkreg)

Coregistro ciascuna PET alla rispettiva risonanza xxx:
coregister
number of subjects:1
which option: coregister & reslice
target image: xxx_soggetto.img
source image: PET_soggetto.img
other images for subj 1: Done
→ rxPET_soggetto.img

Check coregistrazione

Creo il template PET customizzato:
- Utilizzo il template di VI generazione (template_genVI_2x2x2.img), creato x studi SPECT (\\file-server\IRCCS3\Documenti\Uffici\LENITEM\Utenti\Anna\SPECT\hippmask.doc) e descritto nei due lavori su JNeurol, come template di risonanza (è in convenzione neurologica!)
- normalizzo lo stack xxxx (quello flippato all'inizio, e che ora è in convenzione neurologica) al template di VI generazione (template_genVI.hdr) usando SPM2:

Setto i defaults x fare normalizzazione lineare + non lineare:
parameter estimation:
weight template when registering: don't weight
weight source images when registering: don't weight sources
cutoff: 25mm
nonlinear regularization: medium regularization
n. nonlinear iterations: 12
writing normalized:
preserve what: preserve concentration (suggerimento Leighton)
bounding box: template
voxel size: 2x2x2
interpolation method: trilinear interpolation
way to wrap images: no wrap

→ wxxx_soggetto.img

- controllo che la normalizzazione abbia funzionato (checkreg template e wxxx) ed escludo gli eventuali soggetti per cui non ha funzionato
- normalizzo con SPM2 la PET coregistrata alla rispettiva MR utilizzando i parametri ottenuti dalla normalizzazione della MR al template di V1 generazione:

Normalize
Write normalized only
Parameter set: xxxx_soggetto_sn.mat (parametri ottenuti dalla normalizzazione della xxxx_soggetto al template di V1 generazione)
Image to write normalized: rxPET_soggetto.mnc
→ wrxPET_soggetto.img

(è meglio non determinare i parametri delle RM e scrivere direttamente sulle PET, ma avere anche le wxxxx per avere la possibilità di controllare che la normalizzazione delle RM abbia funzionato)

- faccio la media delle wrPET → mean_wrxPCL_NC.img

Normalizzazione a due stadi al template PET (mean_wrPCL_NC.img):

- Smoothing a 10mm FWHM → immagini_sr
- **Normalizzazione lineare** → immagini_wr

Settare i defaults:
Defaults
Spatial normalization
Defaults for parameter estimation
Weight template when registering? Don't weight (*applico la rimozione dell'attività facciale dopo*)
Weight source images when registering? Don't weight sources
Cutoff: affine only
nonlinear regularization: Medium regularization
n. nonlinear iterations: 12 nonlinear iterations
Defaults for writing normalized
Preserve what: preserve concentration
Bounding box: template
Voxel sizes: 2x2x2
Way to wrap images: no wrap
Normalize
Which option: determine parameters and write normalized
mask object brain when registering? Don't mask object
template: mean_wrPCL_NC.img
subj 1: image to determine parameters : imagine_s
subj 1: image to write normalized : imagine_r
subj 2: … …
normalization type: default normalization
interpolation method: trilinear interpolation

- Controllare che la registrazione lineare abbia funzionato (con check reg)
- **Rimozione dell'attività facciale e dello scalpo** → immagini_awr

Toolbox
Masks
Select operations: weight or mask images
Select mask/weight file: mascheravolta, un po' più larga della brainmask e binaria)
Select images: immagini_wr

- Smoothing a 10mm FWHM → immagini_sawr
- **Normalizzazione non lineare** → immagini_wawr

Settare i defaults:

Defaults
  Spatial normalization
  Defaults for parameter estimation
    Weight template when registering? Don't weight (*ho già applicato la rimozione dell'attività facciale*)
    Weight source images when registering? Don't weight sources
    Cutoff: 25mm (equivale a 7x9x7 funzioni di base)
    nonlinear regularization: Medium regularization
    n. nonlinear iterations: 12 nonlinear iterations
  Defaults for writing normalized
    Preserve what: preserve concentration
    Bounding box: template
    Voxel sizes: 2x2x2
    Way to wrap images: no wrap

Normalize
  Which option: determine parameters and write normalized
  mask object brain when registering? Don't mask object
    template: mean_wrPCL_NC.img
    subj 1: image to determine parameters : imagine_sawr
    subj 1: image to write normalized : imagine_awr
    subj 2: … …
  normalization type: default normalization
  interpolation method: trilinear interpolation

- Controllare che la registrazione non lineare abbia funzionato (con check reg)
- **Prescaling: image_zwaw**
- **smoothing**

Smooth
  12mm FWHM
  Select scans: output della normalizzazione
  -> szwawrx IMAGES

SCALING

*** usare lo scaling al cervelletto!

Lo scaling può essere fatto in 3 modalità diverse:

1) creando un file con le perfusioni medie (cervelletto o wholebrain) da usare come nuisance nell'analisi statistica

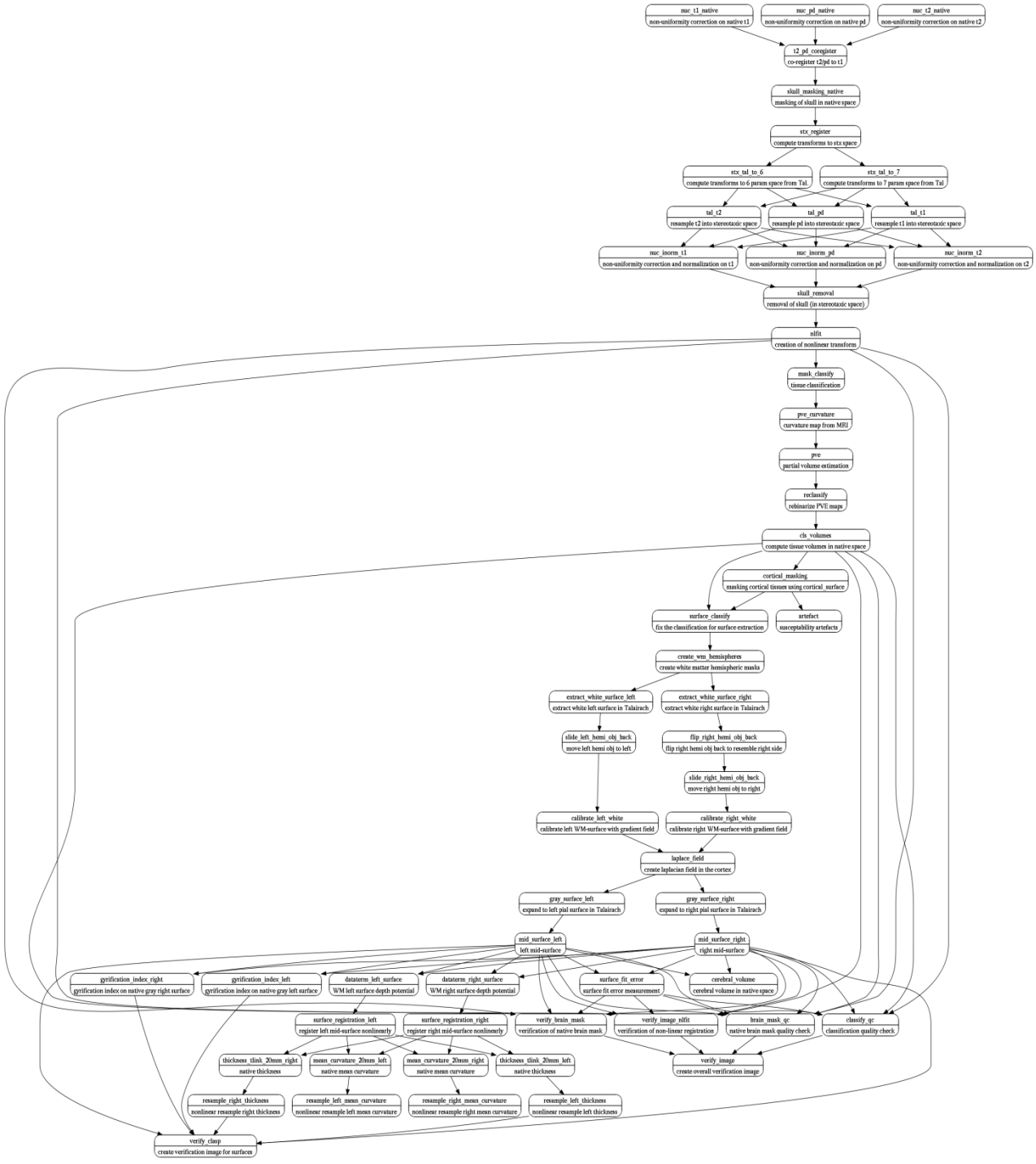
Usando "mask.m", matlab function di Leighton sulle immagini waw:
Toolbox
  Masks
  Select operations: Images(mean)
    Select mask/weight file: brainmask.img (cerebellum.img)
    Select images: immagini_wawr
    Results file: mean_brain (mean_cbllum)

*** Il file risultante va usato come nuisance nelle statistiche (digitando solo mean_brain o mean_cbllum)
*** nell'analisi non usare il proportional scaling, ma "no global normalisation"

2) usando nell'analisi statistica uno scaling proporzionale (ovvero scalando per la perfusione media su tutto il cervello)
*** nell'analisi usare "proportional scaling" invece che "no global normalisation"

3) creando le immagini scalate (zwaw*) prima dell'analisi statistica:
Toolbox
Masks
Select operations: prescale images to mean in mask
Select mask/weight file: brainmask.img (cerebellum.img)
Select images: immagini_wawr
*** crea le immagini zwaw, a cui applicare lo smoothing, da usare poi nell'analisi
*** nell'analisi non usare il proportional scaling, ma "no global normalisation"

Appendix B – CIVET Pipeline Description



Detailed ist of CIVET Pipeline steps (i.e. stage name, label and parameters):

Artefact.pm: { name => "artefact",
Artefact.pm: label => "susceptability artefacts",
Artefact.pm: args => ["class_art", "0.15", "4", $skull_mask,
Classify.pm: { name => "mask_classify",
Classify.pm: label => "tissue classification",
Classify.pm: args => ["classify_clean", "-clobber", "-clean_tags", "-mask_source",
Classify.pm: { name => "pve_curvature",
Classify.pm: label => "curvature map from MRI",
Classify.pm: args => ["pve_curvature", "-clobber", $t1_input,

```
Classify.pm:        { name => "pve",
Classify.pm:        label => "partial volume estimation",
Classify.pm:        args => ["pve_script", "-clobber", "-nosubcortical", @extraPVE,
Classify.pm:        { name => "reclassify",
Classify.pm:        label => "rebinarize PVE maps",
Classify.pm:        args => ["discretize_pve", "-clobber", $pve_csf,
Classify.pm:        name => "cls_volumes",
Classify.pm:        label => "compute tissue volumes in native space",
Classify.pm:        args => ["compute_icbm_vols", "-clobber", "-transform", $t1_tal_xfm,
Clean_Scans.pm:           { name => "nuc_inorm_${type}",
Clean_Scans.pm:           label => "non-uniformity correction and normalization on ${type}",
Clean_Scans.pm:           args => ["nuc_inorm_stage", $input, $output, "${regModel}_mask.mnc",
Cortex_Mask.pm:       { name => "cortical_masking",
Cortex_Mask.pm:       label => "masking cortical tissues using cortical_surface",
Cortex_Mask.pm:       args => ["cortical_mask", $cls, $cortex, $skull_mask, $brain_mask ],
Cortical_Measurements.pm:       { name => "thickness_${tmethod}_${tkernel}mm_left",
Cortical_Measurements.pm:       label => "native thickness",
Cortical_Measurements.pm:       args => ["cortical_thickness", "-${tmethod}", "-fwhm", ${tkernel},
Cortical_Measurements.pm:       { name => "thickness_${tmethod}_${tkernel}mm_right",
Cortical_Measurements.pm:       label => "native thickness",
Cortical_Measurements.pm:       args => ["cortical_thickness", "-${tmethod}", "-fwhm", ${tkernel},
Cortical_Measurements.pm:        name => "resample_left_thickness",
Cortical_Measurements.pm:        label => "nonlinear resample left thickness",
Cortical_Measurements.pm:        args => ["surface-resample", $surfreg_model, $left_mid_surface,
Cortical_Measurements.pm:        name => "resample_right_thickness",
Cortical_Measurements.pm:        label => "nonlinear resample right thickness",
Cortical_Measurements.pm:        args => ["surface-resample", $surfreg_model, $right_mid_surface,
Cortical_Measurements.pm:         name => "thickness_${tmethod}_${tkernel}mm",
Cortical_Measurements.pm:         label => "native thickness",
Cortical_Measurements.pm:         args => ["objconcat", $left_mid_surface, $right_mid_surface,
Cortical_Measurements.pm:         name => "resample_full_thickness",
Cortical_Measurements.pm:         label => "nonlinear resample full thickness",
Cortical_Measurements.pm:         args => ["objconcat", $left_mid_surface, $right_mid_surface,
Cortical_Measurements.pm:         name => "asymmetry_rms_${tmethod}_${tkernel}mm",
Cortical_Measurements.pm:         label => "asymmetry cortical thickness map",
Cortical_Measurements.pm:         args => ["asymmetry_cortical_thickness", "-clobber", $rsl_left_thickness,
Cortical_Measurements.pm:       name => "lobe_area_left",
Cortical_Measurements.pm:       label => "native lobe area",
Cortical_Measurements.pm:       args => ["lobe_area", "-transform", $t1_tal_xfm, $atlas,
Cortical_Measurements.pm:       name => "lobe_area_right",
Cortical_Measurements.pm:       label => "native lobe area",
Cortical_Measurements.pm:       args => ["lobe_area", "-transform", $t1_tal_xfm, $atlas,
Cortical_Measurements.pm:         name => "lobe_area",
Cortical_Measurements.pm:         label => "native lobe area",
Cortical_Measurements.pm:         args => ["lobe_area", "-transform", $t1_tal_xfm, $atlas,
Cortical_Measurements.pm:       name => "mean_curvature_${tkernel}mm_left",
Cortical_Measurements.pm:       label => "native mean curvature",
Cortical_Measurements.pm:       args => ["mean_curvature", "-fwhm", ${tkernel},
Cortical_Measurements.pm:       name => "mean_curvature_${tkernel}mm_right",
Cortical_Measurements.pm:       label => "native mean curvature",
Cortical_Measurements.pm:       args => ["mean_curvature", "-fwhm", ${tkernel},
Cortical_Measurements.pm:        name => "resample_left_mean_curvature",
Cortical_Measurements.pm:        label => "nonlinear resample left mean curvature",
Cortical_Measurements.pm:        args => ["surface-resample", $surfreg_model, $left_mid_surface,
Cortical_Measurements.pm:        name => "resample_right_mean_curvature",
Cortical_Measurements.pm:        label => "nonlinear resample right mean curvature",
Cortical_Measurements.pm:        args => ["surface-resample", $surfreg_model, $right_mid_surface,
Cortical_Measurements.pm:         name => "mean_curvature_${tkernel}mm",
Cortical_Measurements.pm:         label => "native mean curvature",
```

```
Cortical_Measurements.pm:          args => ["objconcat", $left_mid_surface, $right_mid_surface,
Cortical_Measurements.pm:        name => "gyrification_index_left",
Cortical_Measurements.pm:        label => "gyrification index on native gray left surface",
Cortical_Measurements.pm:        args => ["gyrification_index", "-transform", $t1_tal_xfm,
Cortical_Measurements.pm:        name => "gyrification_index_right",
Cortical_Measurements.pm:        label => "gyrification index on native gray right surface",
Cortical_Measurements.pm:        args => ["gyrification_index", "-transform", $t1_tal_xfm,
Cortical_Measurements.pm:          name => "gyrification_index_full",
Cortical_Measurements.pm:          label => "gyrification index on native gray full surface",
Cortical_Measurements.pm:          args => ["gyrification_index", "-transform", $t1_tal_xfm,
Cortical_Measurements.pm:        name => "cerebral_volume",
Cortical_Measurements.pm:        label => "cerebral volume in native space",
Cortical_Measurements.pm:        args => ["cerebral_volume", $final_classify, $final_callosum,
Linear_Transforms.pm:            { name => "nuc_${type}_native",
Linear_Transforms.pm:            label => "non-uniformity correction on native ${type}",
Linear_Transforms.pm:            args => ["nuc_inorm_stage", $input, $output, "none", $nuc_dist,
$nuc_cycles, $nuc_iters],
Linear_Transforms.pm:          { name => "t2_pd_coregister",
Linear_Transforms.pm:          label => "co-register t2/pd to t1",
Linear_Transforms.pm:          args => ["mritoself", "-clobber", "-nothreshold", "-mi", "-lsq6",
Linear_Transforms.pm:            { name => "t2_pd_coregister",
Linear_Transforms.pm:            label => "co-register t2/pd to t1",
Linear_Transforms.pm:            args => ["mritoself", "-clobber", "-nothreshold", "-mi", "-lsq6",
Linear_Transforms.pm:          { name => "skull_masking_native",
Linear_Transforms.pm:          label => "masking of skull in native space",
Linear_Transforms.pm:          args => ["ln", "-sf", $user_mask, $skull_mask ],
Linear_Transforms.pm:          { name => "skull_masking_native",
Linear_Transforms.pm:          label => "masking of skull in native space",
Linear_Transforms.pm:          args => ["remove_skull", "t1Only", $t1_input, $t2_input,
Linear_Transforms.pm:        { name => "stx_register",
Linear_Transforms.pm:          label => "compute transforms to stx space",
Linear_Transforms.pm:          args => ["multispectral_stx_registration", "-nothreshold",
Linear_Transforms.pm:          { name => "stx_tal_to_6",
Linear_Transforms.pm:          label => "compute transforms to 6 param space from Tal.",
Linear_Transforms.pm:          args => ["talto6", $t1_tal_xfm, $tal_to_6_xfm],
Linear_Transforms.pm:          { name => "stx_tal_to_7",
Linear_Transforms.pm:          label => "compute transforms to 7 param space from Tal",
Linear_Transforms.pm:          args => ["talto7", $t1_tal_xfm, $tal_to_7_xfm],
Linear_Transforms.pm:            { name => "tal_t1",
Linear_Transforms.pm:            label => "resample t1 into stereotaxic space",
Linear_Transforms.pm:            args => ["mincresample", "-clobber", "-transform",
Linear_Transforms.pm:            { name => "tal_t2",
Linear_Transforms.pm:            label => "resample t2 into stereotaxic space",
Linear_Transforms.pm:            args => ["mincresample", "-clobber", "-transform",
Linear_Transforms.pm:            { name => "tal_pd",
Linear_Transforms.pm:            label => "resample pd into stereotaxic space",
Linear_Transforms.pm:            args => ["mincresample", "-clobber", "-transform",
Non_Linear_Transforms.pm:          { name => "nlfit",
Non_Linear_Transforms.pm:          label => "creation of nonlinear transform",
Non_Linear_Transforms.pm:          args => ["best1stepnlreg.pl", "-clobber", "-source_mask", $skull_mask,
Segment.pm:          name => "nlfit_animal",
Segment.pm:          label => "creation of nonlinear transform for ANIMAL segmentation",
Segment.pm:          args => ["best1stepnlreg.pl", "-clobber", "-source_mask", $skull_mask,
Segment.pm:          name => "segment",
Segment.pm:          label => "automatic labelling",
Segment.pm:          args => ["stx_segment", "-clobber", $atlas, "-modeldir", $atlasdir,
Segment.pm:          name => "segment_volumes",
Segment.pm:          label => "label and compute lobe volumes in native space",
Segment.pm:          args => ["compute_icbm_vols", "-clobber", "-transform",
```

```
Segment.pm:          name => "segment",
Segment.pm:          label => "automatic labelling",
Segment.pm:          args => ["lobe_segment", "-clobber", "-modeldir", $atlasdir,
Segment.pm:          name => "segment_volumes",
Segment.pm:          label => "label and compute lobe volumes in native space",
Segment.pm:          args => ["compute_icbm_vols", "-clobber", "-transform",
Segment.pm:         name => "segment_mask",
Segment.pm:         label => "mask the segmentation",
Segment.pm:         args => ["minccalc", "-clobber", "-expr", $seg_mask_expr,
Skull_Masking.pm:         { name => "skull_removal",
Skull_Masking.pm:         label => "removal of skull (in stereotaxic space)",
Skull_Masking.pm:         args => ["remove_skull", $maskType, $t1_input, $t2_input,
Surface_Fit.pm:         { name => "surface_classify",
Surface_Fit.pm:         label => "fix the classification for surface extraction",
Surface_Fit.pm:         args => ["surface_fit_classify", $cls_correct, $pve_wm, $pve_csf,
Surface_Fit.pm:         { name => "create_wm_hemispheres",
Surface_Fit.pm:         label => "create white matter hemispheric masks",
Surface_Fit.pm:         { name => "extract_white_surface_left",
Surface_Fit.pm:         label => "extract white left surface in Talairach",
Surface_Fit.pm:         args => ["extract_white_surface", $wm_left_centered,
Surface_Fit.pm:         { name => "extract_white_surface_right",
Surface_Fit.pm:         label => "extract white right surface in Talairach",
Surface_Fit.pm:         args => ["extract_white_surface", $wm_right_centered,
Surface_Fit.pm:         { name => "slide_left_hemi_obj_back",
Surface_Fit.pm:         label => "move left hemi obj to left",
Surface_Fit.pm:         args => ["transform_objects", $white_surf_left_prelim, $slide_left_xfm,
Surface_Fit.pm:         { name => "flip_right_hemi_obj_back",
Surface_Fit.pm:         label => "flip right hemi obj back to resemble right side",
Surface_Fit.pm:         args => ["transform_objects", $white_surf_right_prelim,
Surface_Fit.pm:         { name => "slide_right_hemi_obj_back",
Surface_Fit.pm:         label => "move right hemi obj to right",
Surface_Fit.pm:         args => ["transform_objects", $white_surf_right_prelim_flipped, $slide_right_xfm,
Surface_Fit.pm:         { name => "calibrate_left_white",
Surface_Fit.pm:         label => "calibrate left WM-surface with gradient field",
Surface_Fit.pm:         args => ["calibrate_white", $t1_tal_mnc, $final_classify,
Surface_Fit.pm:         { name => "calibrate_right_white",
Surface_Fit.pm:         label => "calibrate right WM-surface with gradient field",
Surface_Fit.pm:         args => ["calibrate_white", $t1_tal_mnc, $final_classify,
Surface_Fit.pm:         { name => "laplace_field",
Surface_Fit.pm:         label => "create laplacian field in the cortex",
Surface_Fit.pm:         args => ["make_asp_grid", $skel_csf, $left_hemi_white_calibrated,
Surface_Fit.pm:         { name => "gray_surface_left",
Surface_Fit.pm:         label => "expand to left pial surface in Talairach",
Surface_Fit.pm:         args => ["expand_from_white", $final_classify,
Surface_Fit.pm:         { name => "gray_surface_right",
Surface_Fit.pm:         label => "expand to right pial surface in Talairach",
Surface_Fit.pm:         args => ["expand_from_white", $final_classify,
Surface_Fit.pm:         name => "mid_surface_left",
Surface_Fit.pm:         label => "left mid-surface",
Surface_Fit.pm:         args => ["average_surfaces", $mid_surface_left, "none", "none",
Surface_Fit.pm:         name => "mid_surface_right",
Surface_Fit.pm:         label => "right mid-surface",
Surface_Fit.pm:         args => ["average_surfaces", $mid_surface_right, "none", "none",
Surface_Fit.pm:         name => "surface_fit_error",
Surface_Fit.pm:         label => "surface fit error measurement",
Surface_Fit.pm:         args => ["surface_qc", $final_classify, $wm_left_centered,
Surface_Fit.pm:          name => "white_surface_full",
Surface_Fit.pm:          label => "white surface full",
Surface_Fit.pm:          args => ["objconcat", $left_hemi_white_calibrated, $right_hemi_white_calibrated,
```

```
Surface_Fit.pm:          name => "gray_surface_full",
Surface_Fit.pm:          label => "gray surface full",
Surface_Fit.pm:          args => ["objconcat", $gray_surface_left, $gray_surface_right,
Surface_Fit.pm:          name => "mid_surface_full",
Surface_Fit.pm:          label => "mid surface full",
Surface_Fit.pm:          args => ["objconcat", $mid_surface_left, $mid_surface_right,
Surface_Register.pm:          name => "dataterm_left_surface",
Surface_Register.pm:          label => "WM left surface depth potential",
Surface_Register.pm:          args => ["depth_potential", "-alpha", "0.05", "-depth_potential",
Surface_Register.pm:          name => "dataterm_right_surface",
Surface_Register.pm:          label => "WM right surface depth potential",
Surface_Register.pm:          args => ["depth_potential", "-alpha", "0.05", "-depth_potential",
Surface_Register.pm:          name => "surface_registration_left",
Surface_Register.pm:          label => "register left mid-surface nonlinearly",
Surface_Register.pm:          args => ["bestsurfreg.pl", "-clobber", "-min_control_mesh", "80",
Surface_Register.pm:          name => "surface_registration_right",
Surface_Register.pm:          label => "register right mid-surface nonlinearly",
Surface_Register.pm:          args => ["bestsurfreg.pl", "-clobber", "-min_control_mesh", "80",
Surface_Register.pm:          name => "surface_resample_left_white",
Surface_Register.pm:          label => "resample left white surface",
Surface_Register.pm:          args => [ "sphere_resample_obj", "-clobber", $left_white_surface,
Surface_Register.pm:          name => "surface_resample_right_white",
Surface_Register.pm:          label => "resample right white surface",
Surface_Register.pm:          args => [ "sphere_resample_obj", "-clobber", $right_white_surface,
Surface_Register.pm:          name => "surface_resample_left_gray",
Surface_Register.pm:          label => "resample left gray surface",
Surface_Register.pm:          args => [ "sphere_resample_obj", "-clobber", $left_gray_surface,
Surface_Register.pm:          name => "surface_resample_right_gray",
Surface_Register.pm:          label => "resample right gray surface",
Surface_Register.pm:          args => [ "sphere_resample_obj", "-clobber", $right_gray_surface,
Surface_Register.pm:          name => "surface_resample_left_mid",
Surface_Register.pm:          label => "resample left mid surface",
Surface_Register.pm:          args => [ "sphere_resample_obj", "-clobber", $left_mid_surface,
Surface_Register.pm:          name => "surface_resample_right_mid",
Surface_Register.pm:          label => "resample right mid surface",
Surface_Register.pm:          args => [ "sphere_resample_obj", "-clobber", $right_mid_surface,
VBM.pm:            name => "VBM_cls_masked",
VBM.pm:            label => "VBM masking of classified image",
VBM.pm:            args => ["mincmath", "-clobber", "-mult", $cls, $brain_mask,
VBM.pm:          name => "VBM_smooth_${volumeFWHM}_csf",
VBM.pm:          label => "CSF map for VBM",
VBM.pm:          args => ["smooth_mask", "-clobber", "-binvalue", 1, "-fwhm",
VBM.pm:          name => "VBM_smooth_${volumeFWHM}_wm",
VBM.pm:          label => "WM map for VBM",
VBM.pm:          args => ["smooth_mask", "-clobber", "-binvalue", 3, "-fwhm",
VBM.pm:          name => "VBM_smooth_${volumeFWHM}_gm",
VBM.pm:          label => "GM map for VBM",
VBM.pm:          args => ["smooth_mask", "-clobber", "-binvalue", 2, "-fwhm",
VBM.pm:            name => "VBM_smooth_${volumeFWHM}_wm_sym",
VBM.pm:            label => "CSF symmetry map for VBM",
VBM.pm:            args => ["asymmetry_map", "-clobber", $smooth_wm, $smooth_wm_sym],
VBM.pm:            name => "VBM_smooth_${volumeFWHM}_gm_sym",
VBM.pm:            label => "CSF symmetry map for VBM",
VBM.pm:            args => ["asymmetry_map", "-clobber", $smooth_gm, $smooth_gm_sym],
VBM.pm:            name => "VBM_smooth_${volumeFWHM}_csf_sym",
VBM.pm:            label => "CSF symmetry map for VBM",
VBM.pm:            args => ["asymmetry_map", "-clobber", $smooth_csf, $smooth_csf_sym],
Verify_Image.pm:      { name => "verify_brain_mask",
Verify_Image.pm:      label => "verification of native brain mask",
```

```
Verify_Image.pm:      args => [ "mincresample", "-clobber", "-byte", "-like", $t1_tal_final,
Verify_Image.pm:      { name => "brain_mask_qc",
Verify_Image.pm:      label => "native brain mask quality check",
Verify_Image.pm:      args => [ "brain_mask_qc", $skull_mask_native, "${lin_model}_mask.mnc",
Verify_Image.pm:      { name => "verify_image_nlfit",
Verify_Image.pm:      label => "verification of non-linear registration",
Verify_Image.pm:      args => [ "mincresample", "-clobber", "-like", $t1_tal_final,
Verify_Image.pm:      { name => "classify_qc",
Verify_Image.pm:      label => "classification quality check",
Verify_Image.pm:      args => [ "classify_qc", $cls_correct, $classify_info_file ],
Verify_Image.pm:      { name => "verify_image",
Verify_Image.pm:      label => "create overall verification image",
Verify_Image.pm:      args => [ @verifyCmd, @verifyRows ],
Verify_Image.pm:        { name => "verify_clasp",
Verify_Image.pm:        label => "create verification image for surfaces",
Verify_Image.pm:        args => [ "verify_clasp", $gray_surface_left, $gray_surface_right,
```