



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:

D3.2 Database Implementation and Performance Report

Due date of deliverable: month 32

Actual submission date:

Start date of project: January 1st, 2008; Duration: 36 months

Organisation name of lead contractor for this deliverable: **P2** Prodema Informatics AG,
Switzerland

Revision: Version 4

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	
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)	CO

1 Contents

2	Executive Summary	3
3	Introduction.....	3
4	Data Management Requirements in neuGRID.....	4
4.1	Type of Data.....	4
4.2	Source of Data	4
4.3	Data Access.....	5
5	Design of the neuGRID Data Management Infrastructure.....	5
5.1	The Loris Database.....	5
5.2	The Loris-X Database	6
5.3	Data Dictionary.....	7
5.4	Integration within the NeugGrid SOA Environment.....	8
5.4.1	CAS Authentication	10
5.4.2	Grid Storage.....	11
5.4.3	User Interface.....	12
5.4.4	Querying Service	13
5.4.5	Image Import – Pseudonymization Service.....	15
6	Implementation.....	16
7	Report on the current status of the database.....	17
8	Conclusion	18
9	Appendix: Loris-X End Users’ Manual (Version 0.2.3).....	20
9.1	Introduction.....	22
9.2	Loris-X interface.....	22
9.2.1	Browsing by subject	23
9.2.2	Browsing by images (=Quality control interface).....	25
9.2.3	Querying.....	26
9.2.4	Upload of clinical data.....	28
9.3	Viewing.....	29
9.3.1	JIV	30
9.3.2	JIV commands.....	30
9.3.3	Pop-up menu.....	31
9.4	Quality control.....	32
9.4.1	Quality control interface	32

2 Executive Summary

Work package 3 of neuGRID is responsible for the creation and deployment of a database system that supports the neuGRID infrastructure. This task included the creation of a data dictionary that was reported on in D3.1, as well as the database development and documentation activity which is the subject of this deliverable.

This document takes a result-centric approach where our goal is to report on the deployed, integrated “Loris-X” database system as well as to describe design decisions and in particular how these relate to requirements, reuse of concepts or software and integration goals.

The neuGRID database implementation is based on the integration of the pre-existing Loris database with the generic services in neuGRID. To make this integration possible, a number of improvements have been carried out along a new branch of the Loris code, called Loris-X. We present in detail the architectural and design choices we made that led to the current deployment of Loris-X in the neuGRID virtual organisation.

Loris-X was deployed in the last quarter of 2009 and has undergone a number of iterative improvements since then. In this deliverable we present the current deployment status with some indicative benchmarks as well as the end-user’s manual of the database system.

3 Introduction

The neuGRID project undertook the development of a software suite that supports researchers in analyzing large samples of digitized medical images (e.g., Magnetic Resonance Imaging (MRI) scans of the human brain) and related clinical information using advanced image processing algorithms, pipelines and workflows.

The defining characteristics of the neuGRID software suite have been determined by an extensive user requirements elicitation procedure and can be summarized around a few main features, such as:

- a service oriented architecture that supports the modularity, the abstraction and reusability of the neuGRID components
- a grid middleware backbone that supports the sharing of storage capacity and computing power
- a database for the management of images and related clinical data as well as associated meta-information such as, e.g quality control data
- a set of generic services that in part provide abstraction layers on top of functional components (query service, Glueing service), and in part contribute to meeting functional requirements (provenance service, workflow service)
- image processing algorithms and pipelines

The neuGRID requirements, the overall architecture, the design of most of the components is reported in other deliverables such as D9.2, D5.1, D6.1, D7.1, D8.1; this deliverable focuses on the development of the database for MRI images, clinical information and related metadata.

4 Data Management Requirements in neuGRID

4.1 Type of Data

The neuGRID system supports the storage, annotation, quality control and processing of brain MRI images as its main functionality. Additional requirements relate to the secure and private handling of information; manipulation, intelligent execution, planning and optimization of image processing workflows; recording and management of provenance information related to image processing. To understand the overall picture and details as far as requirements defining the neuGRID domain, the reader is kindly directed to consult D9.2, User Requirements Specification.

The universe of discourse for neuGRID is that of 3D MRI brain images with associated data used for analyses and interpretation. This includes acquisition parameters, image characteristics, attributes of the particular examination, attributes that define the subject (patient), related clinical data that has been taken of the subject in the context of the study and additional information establishing context like temporal data to group and interpret follow-up acquisitions. Furthermore, this information is enhanced by quality control (QC) assessment and extended by the results of various processing algorithms and workflows.

In addition to the data described above which are always associated with a particular subject, one also must take into consideration additional information that requires persistence, such as information about users, source of QC data, phantom images, and the very description of the syntax and semantics of the study at hand.

4.2 Source of Data

Users of the neuGRID system are expected to use the neuGRID infrastructure for the processing and sharing of their own research collections of images and related clinical data. To this end, the neuGRID system specifies a protocol for entering two distinct types of data: image data (e.g., MRI scans), and clinical data (typically tabular, scalar data). We refer here to these data types as *unstructured* and *structured* data, respectively. NeuGRID provides an image import endpoint, where users may upload compressed folders of DICOM sequences and a separate one for import of clinical data in XML format.

For reference, and especially for testing and validating purposes, we consider some pre-existing MRI studies as the data source for neuGRID. In particular, the system has been tested with datasets that have been collected as part of the U.S. Alzheimer's Disease Neuroimaging Initiative (ADNI) study¹ and those from the FP6-funded AddNeuroMed².

These pre-existing datasets have been chosen for numerous benchmarking purposes as they provide a good quality and relatively homogeneous cohort that has been used multiple times for data challenges on the neuGRID grid infrastructure, processing hundreds of images by similar workflows simultaneously.

¹ Mueller SG, Weiner MW, Thal LJ, Petersen RC, Jack CR, Jagust W, Trojanowski JQ, Toga AW, Beckett L.: Ways toward an early diagnosis in Alzheimer's disease: The Alzheimer's Disease Neuroimaging Initiative (ADNI), *Alzheimer's & Dementia* 1:55-66, 2005

² Lovestone S, Francis P, Strandgaard K.: Biomarkers for disease modification trials - the Innovative Medicines Initiative and AddNeuroMed, *J Nutr Health Aging* 2007; 11(4):359-361.

To make use of this benchmark dataset, a batch upload functionality was created especially for populating the image data content of the image database.

4.3 Data Access

The neuGRID data access requirements have been thoroughly analysed and documented, along with their impact on ethical, privacy and security issues. Essential requirements define how only properly authenticated neuGRID users may access the infrastructure at all, and may access data that they have rights to. These requirements are explained in detail in D9.2.

To further ensure the conformance to ethical standards and in particular patient privacy, it is essential for neuGRID that only properly de-identified data is kept in the persistent storage of the system at any time. Supporting such ethical requirement is partly aided by computing infrastructure but always has a human component in the ultimate responsibility of the person entering the data into the system. The neuGRID system supports the de-identification by the existence of the pseudonymization service that rids the DICOM files from any identifying information, but it remains with the human user to provide full guarantee that no improper data has gotten through.

neuGRID maintains the persistent store of 3D MRI DICOM files in storage elements indexed by the grid file catalogue. Access to these files is governed by the authorization model provided by the middleware, which also transparently grants or denies access.

5 Design of the neuGRID Data Management Infrastructure

5.1 The Loris Database

The Loris database system was originally developed at the Montreal Neurological Institute for the NIH-funded MRI Study of Normal Brain Development³, and has since been deployed in a number of research studies, including AddNeuroMed, for the collection, quality control, and maintenance of repositories of medical imaging data and related metadata. Consortium partner Prodemia has continued the development of Loris specifically for neuGRID.

The functional requirements of the database components of the neuGRID software suite are largely met by the Loris software package which was thus chosen to be the basis for the data management infrastructure. In particular, the actual version of Loris available at the start of the project provided the following functionalities needed for neuGRID:

- Persistence of both unstructured (DICOM/MINC images) and structured (image metadata, clinical sensors) information in safe and reliable storage
- Easy to use web interface for interaction with the system
- Quality Control user interfaces for creating and reading quality control information which in turn can be used for selecting and filtering data.
- Query interfaces for selecting data cohorts according to user-defined criteria based on image header or clinical data.
- Storage of associated derived data, such as image files resulting from pipeline processing.

³ Evans AC; Brain Development Cooperative Group. The NIH MRI study of normal brain development. *Neuroimage*, 2006 Mar; 30(1):184-202

- Security and privacy based on definition of projects, appropriate authentication and authorization

The architecture of the Loris database had been dictated by the need that it be self-contained and straightforward to install as a near-complete solution to the set of related tasks defined above. This architectural choice, however, makes it difficult to directly integrate it into a system such as neuGRID's, where separate functionalities are provided by services, the coupling is loose and some previously internal tasks are delegated to external components.

For this reason, the development activity related to the imaging and clinical database has focused around two objectives:

- To augment the Loris database with new functionalities that are dictated by neuGRID's functional requirements
- To modularize the Loris codebase and build in the necessary abstraction layers for the purpose of integration within the neuGRID SOA.

In order to accomplish the above, Prodema undertook a refactoring activity on a new branch of the Loris codebase, codenamed Loris-X. In the following, we shall be referring to Loris-X as the version of Loris that has been particularly adopted for being integrated in a SOA, especially neuGRID.

5.2 The Loris-X Database

The Loris-X database is the new branch of Loris that has been refactored along the lines of the MVC⁴ paradigm that provides the needed modularity for integration into the neuGRID SOA. Accordingly, in the design of Loris-X, the domain model has been promoted to a first-class entity (instead of the approach of Loris, where the domain model is manifested in the database schema), it is defined in terms of an object model that is completely independent of the physical database schema. The views and the controllers that provide access to the data only interact with the object model and the physical database is completely shielded from them.

The persistence of the data is provided by mapping the domain model to a physical database schema and connector. Such a database is usually (but not necessarily) backed by a relational database engine thus the database transactions happen through an Object-Relational Mapper (ORM).

The core entities of the Loris-X domain model are classes of main interest in the neuGRID domain of discourse, such as Subject, Event, Examination, MRI Image and Quality control annotation. These first class entities in turn can be further qualified using attributes that represent particularities of the Subject, Event, etc. These attributes can be of various types and (just as in Loris) are not hardcoded in the domain model; rather they can be defined and configured on the fly. To support this, the Loris-X domain model supports meta-categories that encode the various properties, constraints, names of attributes.

⁴ Model-View-Controller

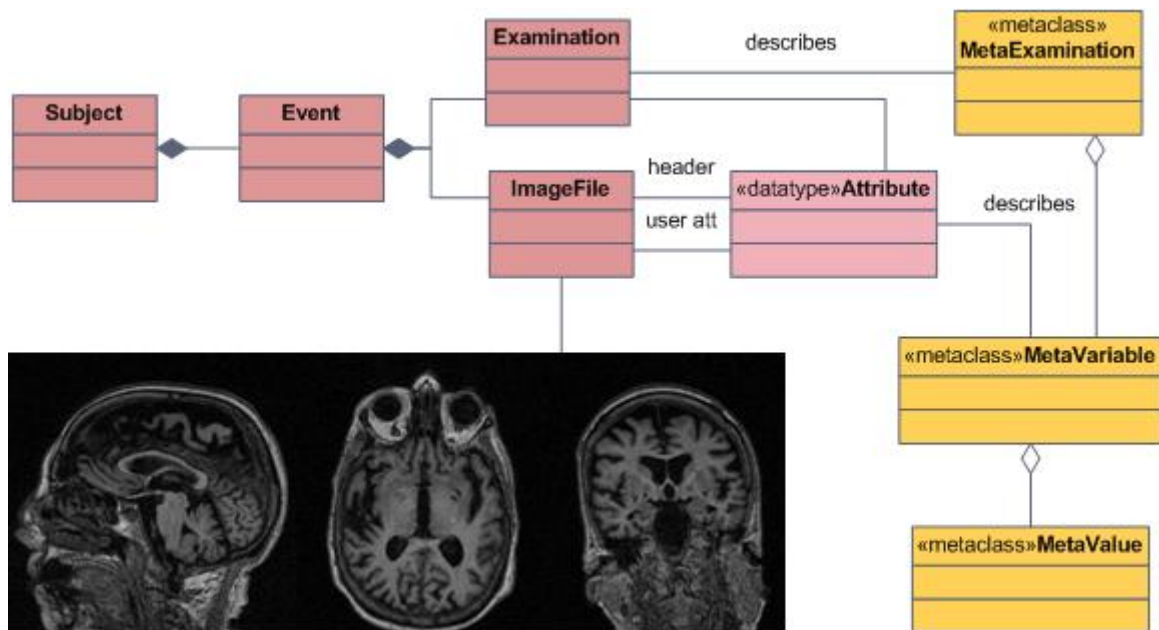


Figure 1 Loris-X domain core entities

In Figure 1, we show the relationship between the core entities of the Loris-X application domain. The main hierarchy (Subject – Event – Examination/ImageFile – Attribute) is mapped to the database. Since this schema is very generic, we use the meta-model to establish the semantics and constraints on the data which in turn allows for proper interpretation (and querying). The approach is very similar to archetype-like data models which are widely used in health informatics.

5.3 Data Dictionary

To store, manage and present information according to the intended semantics, the Loris-X database relies in part on a data dictionary that is assumed to be populated in advance. We note that the lack of, or incompletely populated data dictionary does not render the database unusable; indeed, the users are free to create arbitrary attributes and annotations to capture relevant bits of study information. The Data Dictionary, however, provides the means of maintaining integrity, resolving ambiguity, easing data import, improving query performance and rendering usable views.

The benchmark data dictionary has been created as deliverable D3.1. Therein, we evaluated the ADNI and AddNeuroMed clinical information, derived an alignment of the clinical data attributes and proposed the result as the baseline data dictionary for use in the neuGRID project. We have populated the deployed Loris-X database’s meta-entities with the neuGRID data dictionary and have used this data dictionary for importing the benchmark ADNI data.

The import of structured data into the Loris-X system happens through a simple file upload interface that provides an entry point for an XML parser import component. We created a simple XML schema to define the syntax for data entry that is shared by both meta-information (i.e. populating or augmenting the data model) and real clinical data.

For illustration, a tiny excerpt of the data model input XML document below shows how to establish a variable for patient gender and further link it to the kind of examination that we call ‘Demographics’:

```
<LorisData xmlns='http://www.prodemamedical.com/loris'>
```

```
<examinationtype examid='0' name='Demographics'>
  <variabletype varid='PTGENDER' />
</examinationtype>
<variabletype varid='PTGENDER' type='CHOICE'>
  <description>Participant Gender</description>
  <values>
    <value valid='1' valstring='Male' />
    <value valid='2' valstring='Female' />
  </values>
</variabletype>
</LorisData>
```

5.4 Integration within the Neugrid SOA Environment

The neuGRID system is a service-based infrastructure that exploits the grid paradigm by promoting the sharing and unanticipated reuse of computing and storage elements and software components. Functionalities appear as services establishing a clear separation of responsibilities; loose coupling and service orchestration integrates the needed pieces into a coherent whole for the end-users.

The main concern for the Loris- and Loris-X databases has been to establish sufficient abstraction within the self-contained data store such that it can be integrated against the required service endpoints of the neuGRID system, as well as operate independently.

As far as the middleware integration is concerned, neuGRID has chosen to couple to the generic middleware through a gateway component, a dedicated (possibly virtual) machine running the Pandora gateway software. The neuGRID-configured Pandora gateway provides a service container that seamlessly integrates the deployed services with a number of non-functional requirements such as session governance, authentication, logging, etc.

After a detailed study, we identified the following major integration points with other neuGRID components:

- The database has to be integrated with the common authentication environment, CAS (which itself is aligned with the neuGRID virtual organization (VO) management)
- User interfaces have to be integrated with the unified neuGRID user interface
- A service endpoint has to be exposed for querying the data hosted in Loris-X
- A service endpoint has to be provided for data import into Loris-X
- Integration with generic storage elements, in particular those which are backed up by grid storage, has to be provided via the neuGRID glueing service.

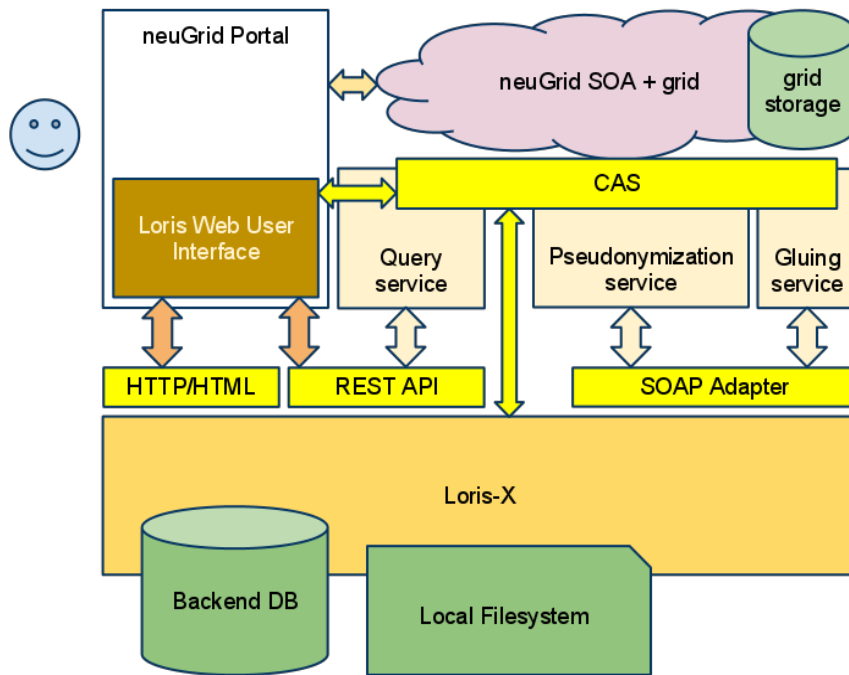


Figure 2 Loris-X Integration Points in neuGRID

In addition to the data access and data creation API defined towards the query service and the pseudonymization service, Loris-X maintains internal links to its own internal storage of both structured and unstructured data. No access to these data stores is provided other than through Loris-X.

In the following sections, we explain in detail our results with respect to integration into the neuGRID SOA.

5.4.1 CAS Authentication

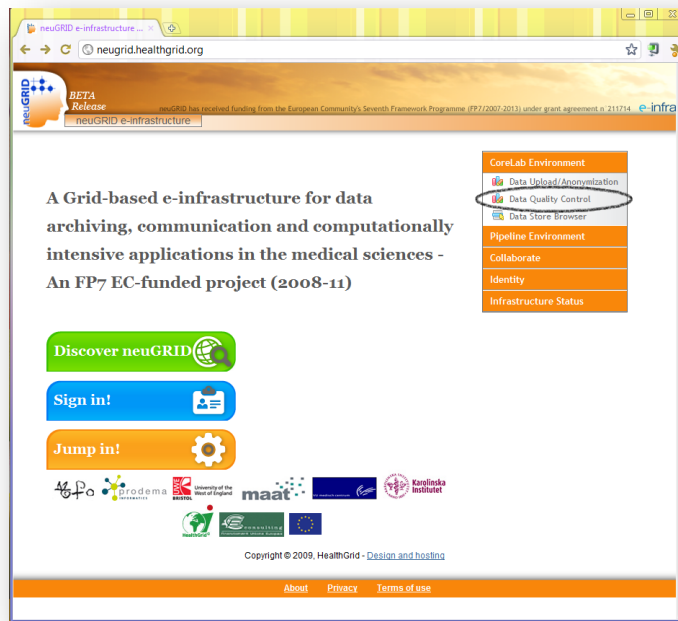


Figure 3 Entering Loris-X through the neuGRID main user interface

The neuGRID infrastructure provides and requests the use of a central, shared authentication infrastructure where the users of the system are managed by the neuGRID virtual organization and authenticate using personal certificates, as it is the de facto standard in the grid community. Further, neuGRID supports proxy certificates and provides online storage of these to make repeated logins, for a limited time, easier. For the detailed description of the neuGRID authentication infrastructure, the reader is advised to consult the relevant deliverable.

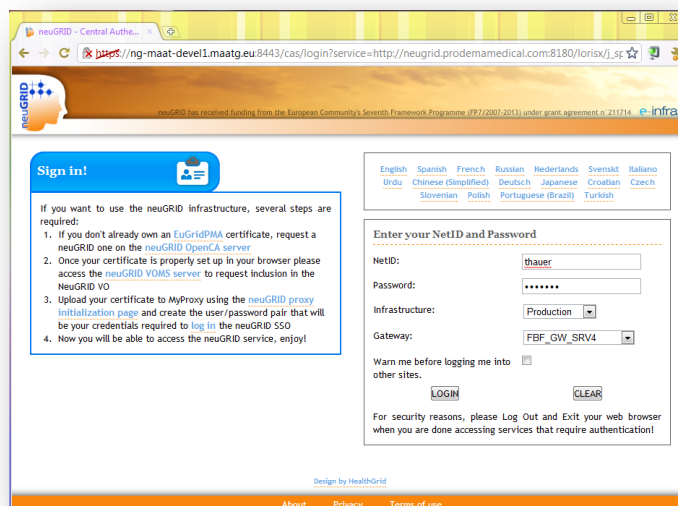


Figure 4 Authentication against the neuGRID CAS server

Authentication in the Loris-X system is provided by an independent authentication module which can support different ways of authentication mechanisms. Integration of Loris-X has

been achieved in two steps. First, an appropriate client plugin that provides CAS authentication was integrated and configured against the neuGRID CAS environment. Then, since Loris-X maintains the concept of users in its own database, a mapping mechanism was created that automatically maps validated neuGRID user-certificate subjects to Loris-X user objects. Thus, the integrated Loris-X implicitly trusts the identity of neuGRID users and opens dedicated users' sessions for them. (During this process, Loris-X also attempts to derive the user's real name from the certificate subject; this can be overwritten or configured differently).

5.4.2 Grid Storage

Loris-X manages both semi-structured and unstructured data that together make up the records of subjects and medical events. For the persistence and management of semi-structured data, Loris-X employs a dedicated database engine that is under Loris-X's governance and is used exclusively by Loris-X. Unstructured data has been chosen to be stored outside of this database, in a file system (as opposed to BLOB records). Loris-X can be configured to use a file system that is co-located with the database or the web server engine, or a more generic file storage remote to it. In neuGRID, some of the unstructured data (e.g. QC preview images) are stored in the former way while most of the data (notably the image sequences) are hosted in remote storage elements under the governance of the neuGRID VO.

The abstraction to access a range of different storage options in a transparent way is provided by the Glueing Service in neuGRID. Note that, this is not the only function of the Glueing service, though, for a detailed description of the Glueing service, the reader is invited to consult deliverable D6.1. The integration with the storage services of Loris-X in neuGRID is thus provided by the interfacing with the Glueing Service.

The Glueing Service (GS) provides generic access to middleware functionalities, let those belong to a grid middleware (like gLite in neuGRID) or represent some other paradigm of computing and storage elements. The Glueing Service links remote components via a generic SOAP webservice protocol, thus in the end a storage client (like Loris-X) interfaces with a GS adapter, which talks to a remote Glueing Service through HTTP/SOAP, which in turn connects to the specific required middleware component using a custom GS connector.

The protocol to invoke the service through the GS adapter was agreed to make use of the SAGA API. The Loris-X integration with the neuGRID middleware, therefore, has been designed to use the SAGA API with appropriate connectors. Those file I/O operations that anticipate the possible use of remote storage have been migrated to an abstract interface layer powered by the SAGA API. The actual functionality can then be verified against a number of available SAGA API connectors. We have validated Loris-X's file I/O operations through this layer using three different connectors: the local connector to reproduce the file storage in co-located hard-drives, the gLite connector that works against gLite SE's and finally the Glueing service connector that uses the translation to the generic webservice SOAP protocol as explained above.

In the neuGRID production environment, the neuGRID VO is configured to make a number of gLite storage elements available. Loris-X is configured to reference the image files that are hosted in this distributed storage, which in part replaces the local storage concepts of Loris.

5.4.3 User Interface

The self-contained Loris software had contained an integrated user interface. This was composed of a web interface providing most end-user functionalities and a command-line interface for particular tools like batch image import. Loris-X, on the other hand, is a component, integrated into a system of many components. From the previous end-user functionalities some are no longer accessed as end-user invoked tasks but instead exposed through an API, while others are still user-accessible through the Loris-X GUI, which is integrated with other neuGRID GUI components.

The following functionalities are offered by the neuGRID Loris-X GUI in the neuGRID deployment:

- Data browsing, primarily to support quality control tasks
- Viewing for the purpose of QC, 2D and 3D image viewing of MRI sequences
- Meta-data uploading to instantiate or modify the dynamic schema and data dictionary used for subject and examination attributes
- Batch uploading of clinical data: subject, event, examination definitions and attributes for subjects, events, examinations and image files
- Querying of clinical data, based on criteria and exporting of the results in various tabulated formats

In the Loris-X design MVC philosophy custom views and controllers are implemented for all relevant Loris-X functionalities, of which the required ones are exposed to the neuGRID integrated UI.

End users typically access the neuGRID system through the integrated neuGRID portal, which serves as the common entry point to the system. It provides the authentication interface, maintains an authenticated session against the neuGRID Pandora gateway, acts as a container for portlets and invokes child functionalities through a hierarchical launch menu.

The above Loris-X functionalities are grouped under the “Data Quality Control” submenu of the Portal dashboard, and the Loris-X user interface is launched as an integrated webservice view in the Portal container. In other words, Loris-X functionalities are not broken down to portlets themselves, but rather are exposed as one integrated webview portlet onto the Loris-X webservice.

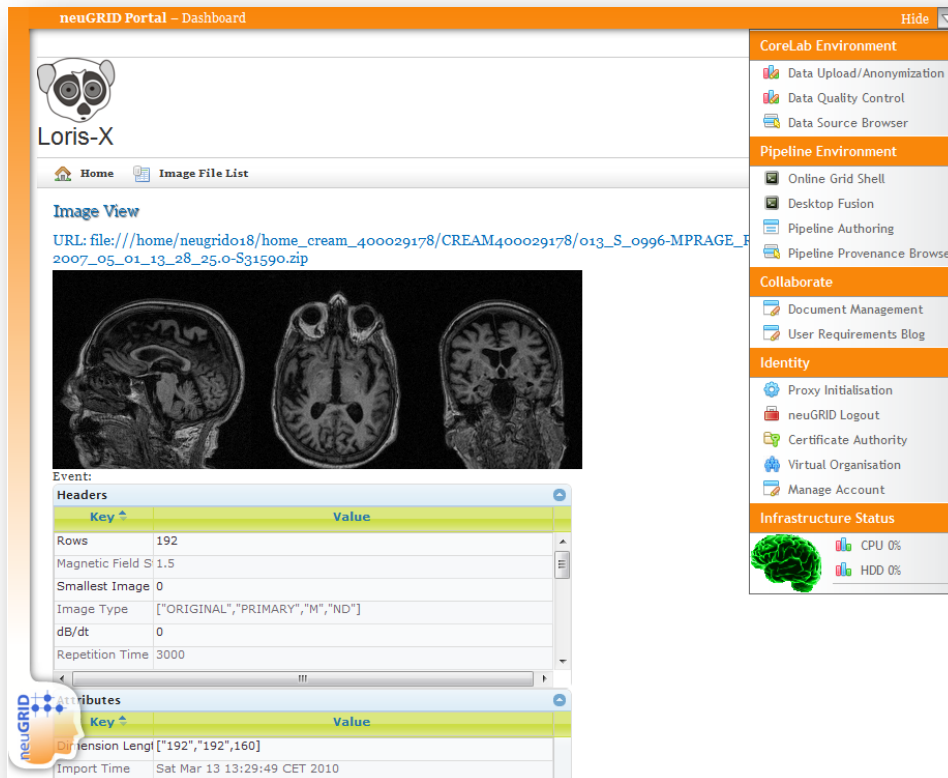


Figure 5 Loris-X UI embedded in the neuGRID Portal Dashboard.

5.4.4 Querying Service

The data that is managed by Loris-X is usually accessed by end-users directly only for the purpose of Quality Control tasks. In other neuGRID use cases, the data is typically requested by the service(s) that support the solution to the use case. For example, in an image processing use case, requesting the images to be processed would be initiated by the component that executes the workflow the user had previously defined.

Thus, unlike what the typical use of the self-contained Loris system had been, the data in Loris-X needed to be exposed both through end-user views as well as programmatic query APIs. To this end, we created a web service layer and Loris-X provides a REST as well as SOAP API to integrate with those components of the SOA that seek access to neuGRID structured data.

There is, notably, a querying service component of the neuGRID SOA whose role is to provide a unified abstract query interface to all data stores in neuGRID, including Loris-X, the provenance database and the service UDDI. The querying service acts as the focal point for structured data access just like a file catalogue would provide access to files, hiding physical layout and schema complexity of the databases it connects to. In the case of Loris-X, the syntax and semantics of the database is represented by the API which encapsulates the information about the domain stored within. Below is part of the schema definition (not including the meta-categories), in wsdl schema that is exposed and facilitates the integration with the query service. Note that, while it is expected that the querying service is the standard data provider endpoint in the neuGRID environment, the Loris-X query API does not discriminate and any properly authenticated entity may connect to it.

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="LorisXSOAPAdaptor"
  targetNamespace="http://www.prodemedical.com/lorisx/schemas/LorisXSOAPAdaptor"
  xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
  xmlns:tns="http://www.prodemedical.com/lorisx/schemas/LorisXSOAPAdaptor"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/">
  <wsdl:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.prodemedical.com/lorisx/schemas/LorisXSOAPAdaptor">
      <xsd:element name="queryForSubjects" type="tns:querySpecification_t"/>
      <xsd:element name="queryForSubjectsResponse" type="tns:queryResponse_t"/>
      <xsd:element name="queryForEvents" type="tns:querySpecification_t"/>
      <xsd:element name="queryForEventsResponse" type="tns:queryResponse_t"/>
      <xsd:element name="queryForImages" type="tns:querySpecification_t"/>
      <xsd:element name="queryForImagesResponse" type="tns:queryResponse_t"/>
      <xsd:element name="startImageImport" type="tns:startImageImport_t"/>
      <xsd:element name="startImageImportResponse" type="tns:startImageImportResponse_t"/>
      <xsd:element name="getImageImportStatus" type="tns:getImageImportStatus_t"/>
      <xsd:element name="getImageImportStatusResponse"
        type="tns:getImageImportStatusResponse_t"/>
      <xsd:complexType name="querySpecification_t">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="1" name="outputColumn"
            type="xsd:string" />
          <xsd:element maxOccurs="unbounded" minOccurs="0" name="constraint"
            type="tns:constraint_t" />
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="constraint_t">
        <xsd:attribute name="columnName" type="xsd:string"/>
        <xsd:attribute name="operator" type="xsd:string" />
        <xsd:attribute name="value" type="xsd:string" />
      </xsd:complexType>
      <xsd:complexType name="queryResponse_t">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="0" name="record"
            type="tns:queryResponseRecord_t" />
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="queryResponseRecord_t">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="1" name="Value"
            type="tns:queryResponseRecordValue_t" />
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="queryResponseRecordValue_t">
        <xsd:simpleContent>
          <xsd:extension base="xsd:string">
            <xsd:attribute name="columnName" type="xsd:string"/>
          </xsd:extension>
        </xsd:simpleContent>
      </xsd:complexType>
      <xsd:complexType name="startImageImport_t">
        <xsd:sequence>
          <xsd:element name="imageUri" type="xsd:string"/>
        </xsd:sequence></xsd:complexType><xsd:complexType name="startImageImportResponse_t">
        <xsd:sequence>
          <xsd:element name="importJobId" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="getImageImportStatus_t">
        <xsd:sequence>
          <xsd:element name="importJobId" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="getImageImportStatusResponse_t">
        <xsd:sequence>
          <xsd:element name="status" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType></xsd:schema>
    </wsdl:types>
    <wsdl:message name="queryForSubjectsResponse">
      <wsdl:part name="parameters" element="tns:queryForSubjectsResponse"/>
    </wsdl:message>
    <wsdl:message name="startImageImportRequest">
      <wsdl:part name="parameters" element="tns:startImageImport"/>
    </wsdl:message>
  </wsdl:definitions>

```

```

</wsdl:message>
<wsdl:message name="queryForImagesRequest">
  <wsdl:part name="parameters" element="tns:queryForImages"/>
</wsdl:message>
<wsdl:message name="getImageImportStatusRequest">
  <wsdl:part name="parameters" element="tns:getImageImportStatus"/>
</wsdl:message>
<wsdl:message name="queryForEventsResponse">
  <wsdl:part name="parameters" element="tns:queryForEventsResponse"/>
</wsdl:message>
<wsdl:message name="getImageImportStatusResponse">
  <wsdl:part name="parameters" element="tns:getImageImportStatusResponse"/>
</wsdl:message>
<wsdl:message name="startImageImportResponse">
  <wsdl:part name="parameters" element="tns:startImageImportResponse"/>
</wsdl:message>
<wsdl:message name="queryForEventsRequest">
  <wsdl:part name="parameters" element="tns:queryForEvents"/>
</wsdl:message>
<wsdl:message name="queryForSubjectsRequest">
  <wsdl:part name="parameters" element="tns:queryForSubjects"/>
</wsdl:message>
<wsdl:message name="queryForImagesResponse">
  <wsdl:part name="parameters" element="tns:queryForImagesResponse"/>
</wsdl:message>
<wsdl:portType name="LorisXSOAPAdaptorPortType">
  <wsdl:operation name="queryForSubjects">
    <wsdl:input message="tns:queryForSubjectsRequest"/>
    <wsdl:output message="tns:queryForSubjectsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="queryForEvents">
    <wsdl:input message="tns:queryForEventsRequest"/>
    <wsdl:output message="tns:queryForEventsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="queryForImages">
    <wsdl:input message="tns:queryForImagesRequest"/>
    <wsdl:output message="tns:queryForImagesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="startImageImport">
    <wsdl:input message="tns:startImageImportRequest"/>
    <wsdl:output message="tns:startImageImportResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getImageImportStatus">
    <wsdl:input message="tns:getImageImportStatusRequest"/>
    <wsdl:output message="tns:getImageImportStatusResponse"/>
  </wsdl:operation>
</wsdl:portType>
</wsdl:definitions>

```

5.4.5 Image Import – Pseudonymization Service

The single entry point for individual data entry of image files for neuGRID is the pseudonymization service. End-users are expected to invoke this service which is capable of receiving digitized image files (DICOM, compressed), and initiates the process of de-identifying, storing and registering the result in the neuGRID data management systems.

The workflow that meets the requirements and is orchestrated by the pseudonymization service had been designed like this:

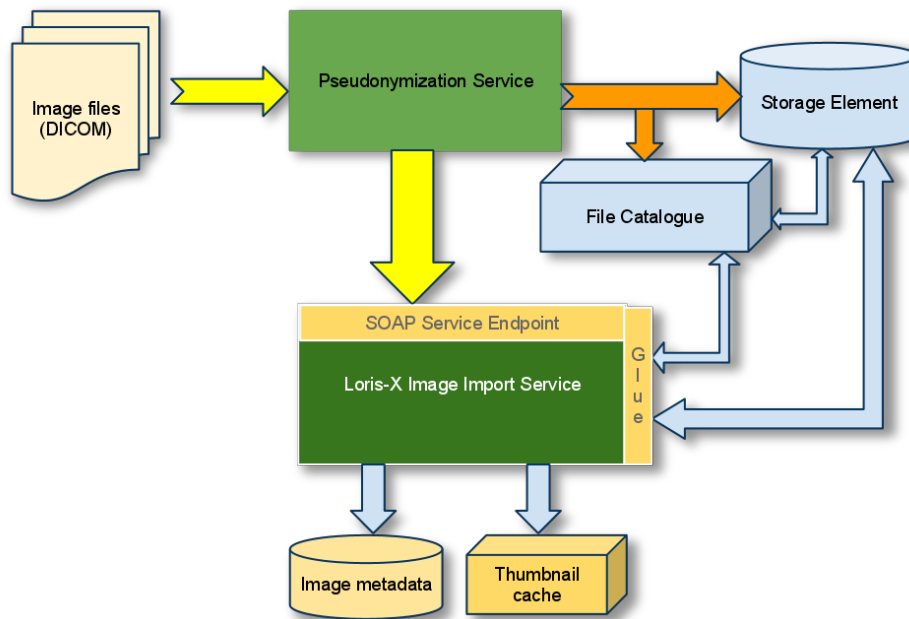


Figure 6 Image Import Workflow

From the point of view of Loris-X, the integration point is the assumption of the existence of a file that has been properly stored and registered in a file catalogue and can be accessed by calling on the appropriate URL. To integrate with the pseudonymization service, Loris-X exposes a SOAP endpoint, whose image import function gets invoked with the logical file name (URL) as parameter. Loris-X then verifies that it has access to the file that needs importing and starts an asynchronous import process, returning a handle for the purpose of querying the status of the import.

The objective of the import process is to properly register the file in the database, to associate it with the existing structured data if needed, to annotate it with the extracted DICOM attributes and to create thumbnail data for quality control views. At the end of this asynchronous process, the requesting client receives a completed status upon status query. Loris-X does not transfer the file, it remains in the same physical location the pseudonymization service referred to originally.

6 Implementation

The implementation of the neuGRID data management has been supported by three pillars: the existing Loris codebase, which is an implementation of most of the needed functionalities; the neuGRID user requirements specifications; and the neuGRID architectural constraints especially related to integration within the neuGRID software suite.

In section 5, we presented some of our design choices that support these requirements. In the current section, we provide a summary of implementation details.

In order to provide the abstraction layers for Loris-X that are needed to expose selected functionalities in a modular way, we engaged in a refactoring effort whereby we migrated the Loris engine to a framework that is built around the MVC modular paradigm. Since much of Loris functionality that's needed to reuse had been written in Java and it was also the most supported API implementation language favoured by other partners, we chose Spring,

the open source application framework for the Java platform, and in turn the rapid development framework Groovy and Grails.

Grails defines the application with clear separation along the model, view, controller and services lines. We migrated and instantiated the Loris domain model in the grails framework, and developed a meta-model for the data dictionary which has been a novel addition to Loris-X. This was necessary because the data dictionary in Loris has been tightly coupled to the particular database engine, a constraint we intended to relax in case of Loris-X.

For the view technology, we maintained the web browser user interface views provided by the combination of HTML, CSS and javascript, like in Loris. The server-side templating engine has been replaced with that of grails and a few novel javascript libraries have been included for modern UI effects, like accordion views, configurable table views, and tree views.

We continue to provide the applet-based 3D MRI viewer, JIV. All the imported images undergo an asynchronous thumbnail generating process whereby data conforming to JIV syntax is generated. This data is in turn served for the viewer applet to allow the user quick 3D navigation within the image, primarily for the purpose of quality control.

For the integration to CAS (see the design above), we migrated to a configurable, generic authentication plugin (acegi, or spring security) that supports a number of different authentication backends, thus extending the existing authentication schemata of Loris with additional ones, CAS included.

As far as the service endpoints are concerned, we have used the controller layer that the grails framework natively provides. Next to the controllers accessed by requests made by web browsers, we created controllers instantiating a REST webservice API and one which is powered by SOAP.

One of the most important features we have gained from migrating to an MVC framework is complete independence from the physical database engine. The mapping to the persistence layer is completely automatic and is provided, in our case, by GORM, the grails object-relational mapping. As a result, we have been able to test Loris-X against different database implementations during development and although we have specified a concrete backend in the final deployment, Loris-X does not depend on it.

The neuGRID deployment relies on a MySQL backend; the database engine is in fact collocated with the webservice engine of the production environment. The Loris-X internal file store which is used for caching and thumbnail generation and storage, is linked to Loris-X as a remotely mounted folder.

7 Report on the current status of the database

During the course of neuGRID, we initially deployed the standalone Loris database to provide accessible storage to the neuGRID benchmark data. The first version of Loris-X available for evaluation was version 0.1, deployed in Prodema's dedicated neuGRID servers in January, 2010. The final deployed version is 0.2.3 with no further version updates anticipated.

Prodema hosts, in its server room, a pair of Dell PowerEdge (PE2950 and PE1950) rack servers to provide databasing support for the neuGRID production environment. The PE1950-1 with its 8 core Xeon E5430@2.66 GHz and 8GB Ram hosts the MySQL databases server as well as the apache + tomcat web server and service container. The PE2950 acts as

the network file store, with a 2TB disk mounted on PE1950 over the intranet. The internet connection to Prodema's servers is a symmetric, 10MB/s service provided by Swisscom.

The database is populated with a data dictionary that includes 811 clinical variable descriptions (MetaVariable, defining name, data type, possible values, etc...) and 47 examination template definitions (MetaExamination, defining the notion of different clinical examinations).

Data import is still ongoing, as explained in the design section; images go through a two-phase import process that gets them fully registered. At the time of this writing, 2589 MRI images are fully registered in the database. Clinical information includes 52245 examination records that are associated to 6481 study events of 1336 different subjects.

The Loris-X user interface has been integrated with the neuGRID portal as explained in section 5.4.3 with the necessary functionalities exposed to users authenticated against the neuGRID VO (**Error! Reference source not found.**). The users use the subject and image browsing interfaces (**Error! Reference source not found.** and **Error! Reference source not found.**) to navigate within the database, and the image viewer interfaces to perform quality control (**Error! Reference source not found.**, **Error! Reference source not found.**). Construction of queries is also possible using directly the Loris-X interface (**Error! Reference source not found.**, **Error! Reference source not found.**), or other interfaces that connect to Loris-X via the querying service.

Query performance has been tested qualitatively. We have carried out a number of iterations for tuning performance. The penalty of a schema-less and object-oriented, mapped model (which is so attractive on the functional side) necessarily rears its head on the querying side. To overcome this, we consciously planned for and dedicated time to develop appropriate indices and materialized views. With the current data population, all queries we tested run within web-time, i.e. within the time which is necessarily taken up by HTTP traffic both ways, request parsing, response construction and client-side rendering. For comparison, with all optimizing features switched off, we estimated this to be around 500-fold speed increase (lower bound).

Storage volume is not of any concern with the current deployment of neuGRID, the current storage provision easily allows for many-fold increase of both the database backend file storage and the file (thumbnail) cache store. If ever needed, installation of extra storage and seamless migration to larger volume is not a technical problem.

We have not experienced any performance issue originating from network bottlenecks. The user base of the neuGRID project itself does not warrant for a more sophisticated networking solution but if the user base of the neuGRID system increased significantly, then network bandwidth stress tests and especially concurrency tests will be needed to evaluate the current setup. The hosting, however, relies on industry standard components, of which we have chosen the ones that are straightforward and accessible and fulfil our requirements. More performant service containers, hosting setups, backend database configurations can be readily available and migration would be trivial as soon as the demand justified the effort and the price.

8 Conclusion

In this deliverable we have reported on the results of database development that has been carried out in work package 3 of neuGRID.

We analyzed the requirements, the constraints and the technologies that neuGRID has committed to as well as the domain requirements that we earlier reported on in the form of the data dictionary deliverable. As a result of this analysis we proposed the neuGRID Loris-X solution that was based on pre-existing Loris technology, novel development that addressed abstraction and modularization and creating suitable integration points so that the MRI database becomes an organic part of the neuGRID SOA.

The main elements of our proposed design were reviewed and explained in detail based on three aspects: the successful migration to the MVC framework, the introduction of metadata entities to manage the contents of the data dictionary and the creation of the required service endpoints.

We showed in detail some of the concrete development and deployment details and through that we demonstrated how the integrated Loris-X meets the functional requirements related to image data import, clinical scalar data import, data viewing and querying and quality control. We analysed some of the non-functional requirements, especially those that relate to the integration challenges like adherence to common authentication of accessing transparent storage elements.

The Loris-X database is currently deployed as part of the neuGRID distributed infrastructure, the details of this deployment we also presented along with some benchmark characteristics that demonstrate that the neuGRID database implementation has been successfully carried out and the resulting deployment is an organic, key part of the neuGRID application suite.

The current version of the Loris-X end user manual is delegated to an Appendix, which gives the interested reader more and finer detail about end-user aspects of the database system.

Database Users Manual

Prodema Medical

0.2.3

List of Abbreviations

DB	Database
MRI	Magnetic resonance imaging
QC	Quality control
CSV	Comma separated value file
Excel	Excel spreadsheet
ODS	OpenDocument spreadsheet
MMSE	Mini mental state examination

9.1 Introduction

Loris-X is a highly configurable database (db) developed by Prodema Medical. Its main purpose is storing images and clinical information. The system is developed to include control mechanisms for optimised multi-site coordination, in particular interactive and web based tools to calculate distance to target, delay of data delivery and quality control. The Loris-X database system covers the entire process from image acquisition to data querying for analysis. Its modular conception and user friendly web interface provides information to a physically distant group of project members. Data management and workflow are transparent as information on all related issues is readily available. The database can be browsed and queried by subjects, images and events.

The system features:

- Multi site – multi center capabilities
- Fully automated data flow
- Possibility of automated data transfer from scanner to repository
- Data archiving and documentation features (on different levels - from study to single file for guaranteed data integrity)
- Automated upload with automated DICOM header verification and scanning protocol recognition
- Automated data organization
- Automated generation of content reports and live statistics on image data quality and project progress.
- Online data entry for instrument data
- Location independent data availability
- Intuitive web interface
- Integration of imaging data with clinical/behavioral data
- Seamless integration with image processing applications
- Intuitive query interface for data download
- 3D visualization of images and image overlay in any web browser

This manual describes the current version of Loris-X (0.2.3). The db is currently under development, which means new features will be added and some other features might change or be replaced. Therefore, this manual might not fully apply for later versions of the db. This manual will be updated in parallel with the development of Loris-X. Thus, make sure you have the manual corresponding to the current version of the db.

9.2 Loris-X interface

Loris-X interface is a web based interface that allows the end user to interact with the database. Figure 1 shows the main menu of the Loris-X interface. As can be observed, there are three options to choose from on this page.

- (1) the database can be browsed by subjects and images
- (2) the database can be queried by subjects, events or images
- (3) Upload clinical data

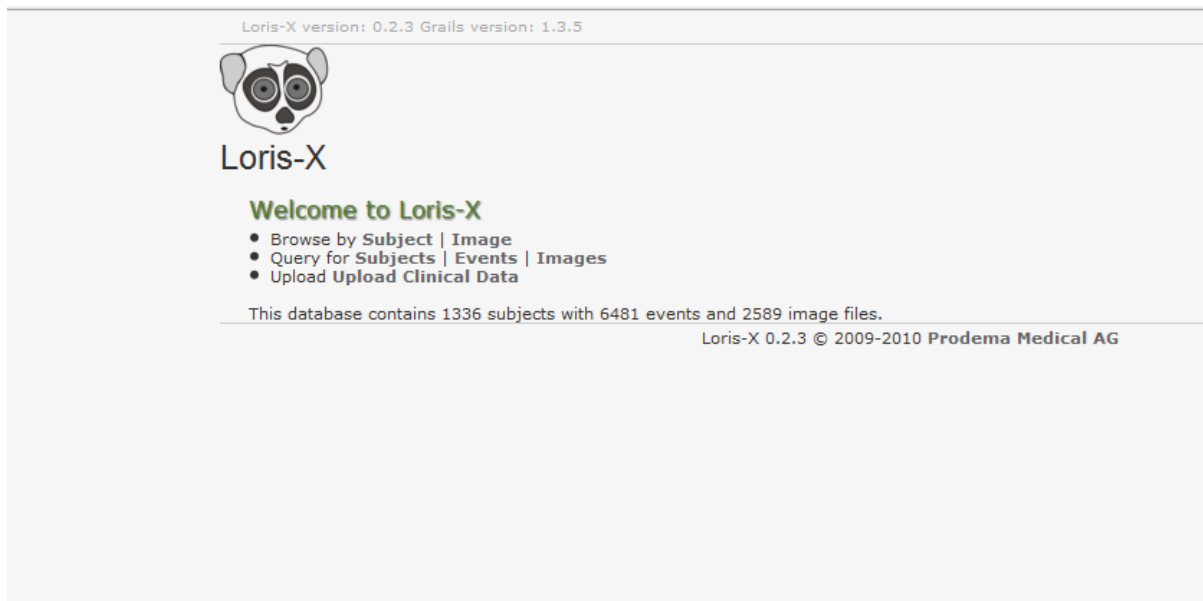


Figure 1: Main menu of Loris-X

9.2.1 Browsing by subject

By entering browsing by subject from the main menu, a new view will appear which enables browsing the entire database for all the subjects it contains (figure 2). Each page that opens contains maximally 10 subjects. Below the subjects list, there is an action bar named next. This action bar will visualize the next set of ten subjects in the database. Another possibility of moving forward in the subjects list is to choose one of the numbers in the list displayed. For example, choosing number seven, the db will move six pages forward in the list (displaying subject 51-60).

The subjects list contains two different columns; namely the column entitled “id” and the column entitled “subject id”. While “id” is the number of order by which the subjects were added to the database, the “subject id” is the identification number of the entered subject.

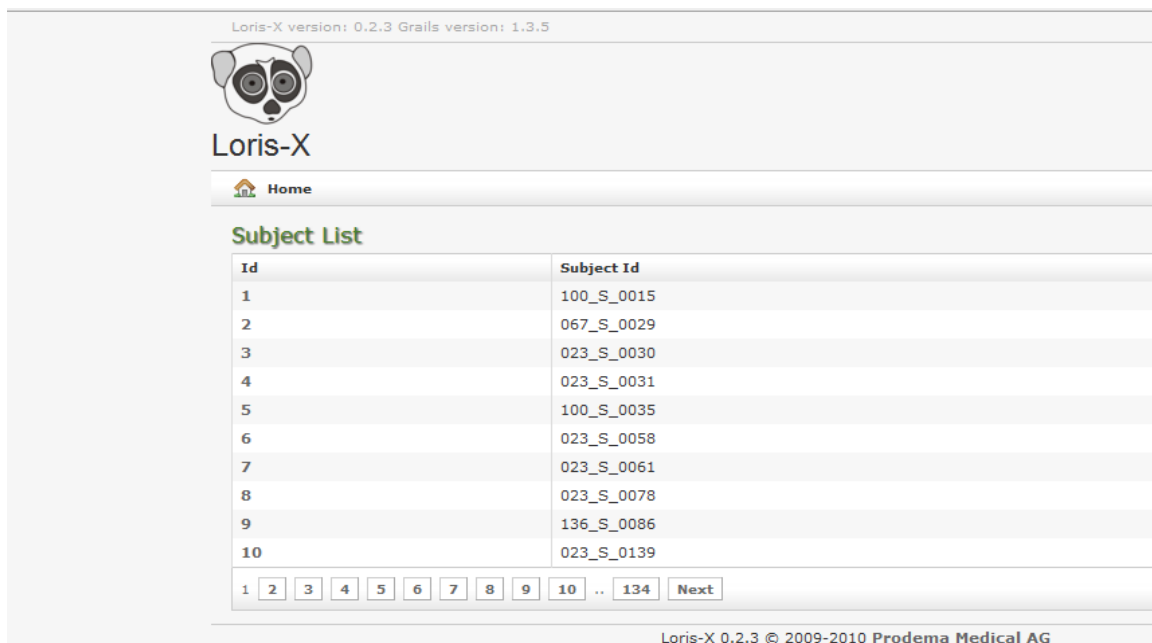


Figure 2: Browsing by subject

By choosing the id corresponding to the subject id of interest a new page will be displayed (figure 3). This view visualizes the event list of the subject of interest.

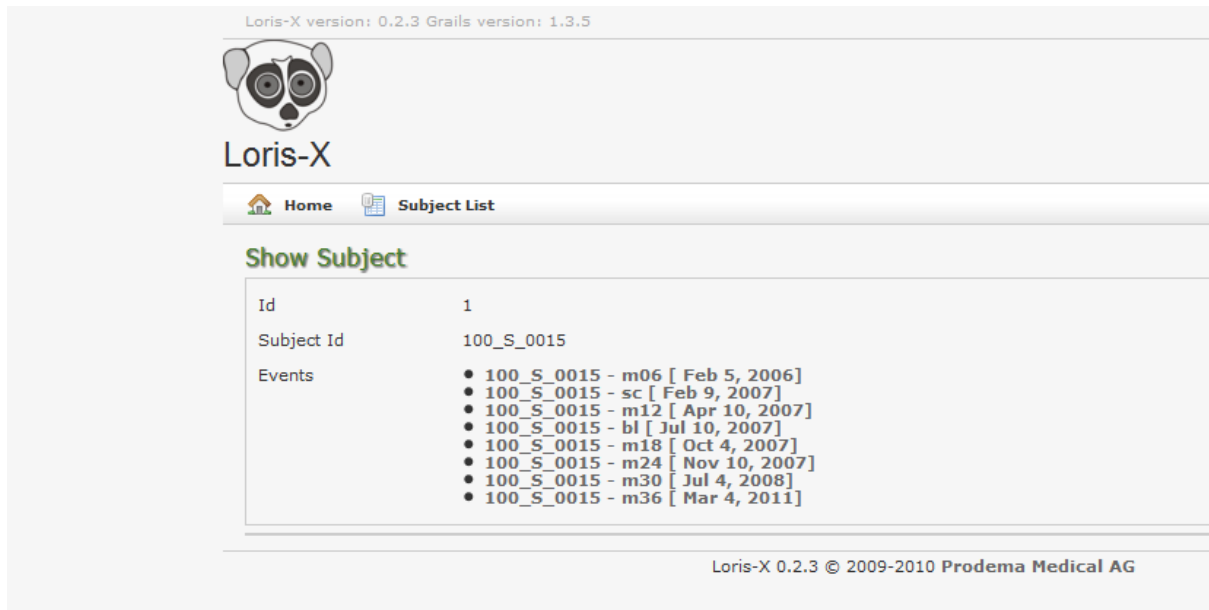


Figure 3: Subject view

The Event list shows the different time points the subject have been examined, for example: screening, baseline and 6 month follow up etc. By entering a page of any event of interest the db moves forward to a view which will displays the information related to this event and all the types of examinations which have been performed at this event (figure 4). Note that an event is not necessary related to only one date, but can refer to information collected at different days. This means that clinical data and image data can be collected at different days but have the same event label such as baseline.

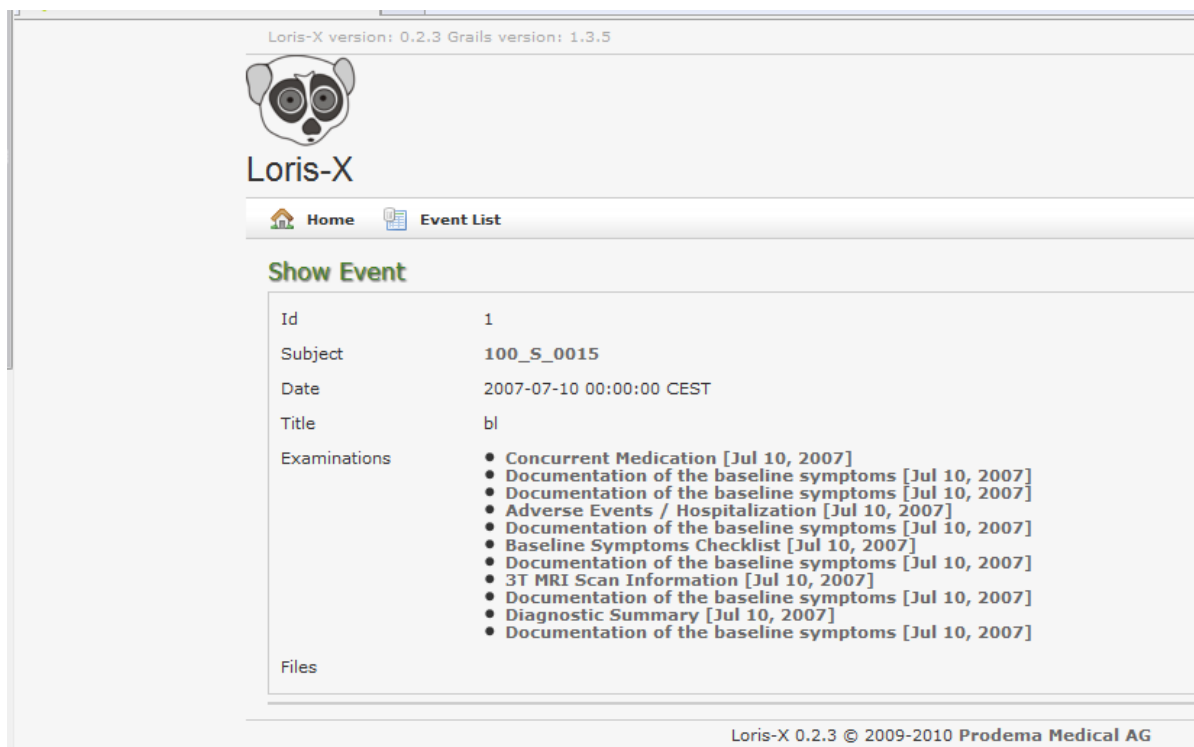
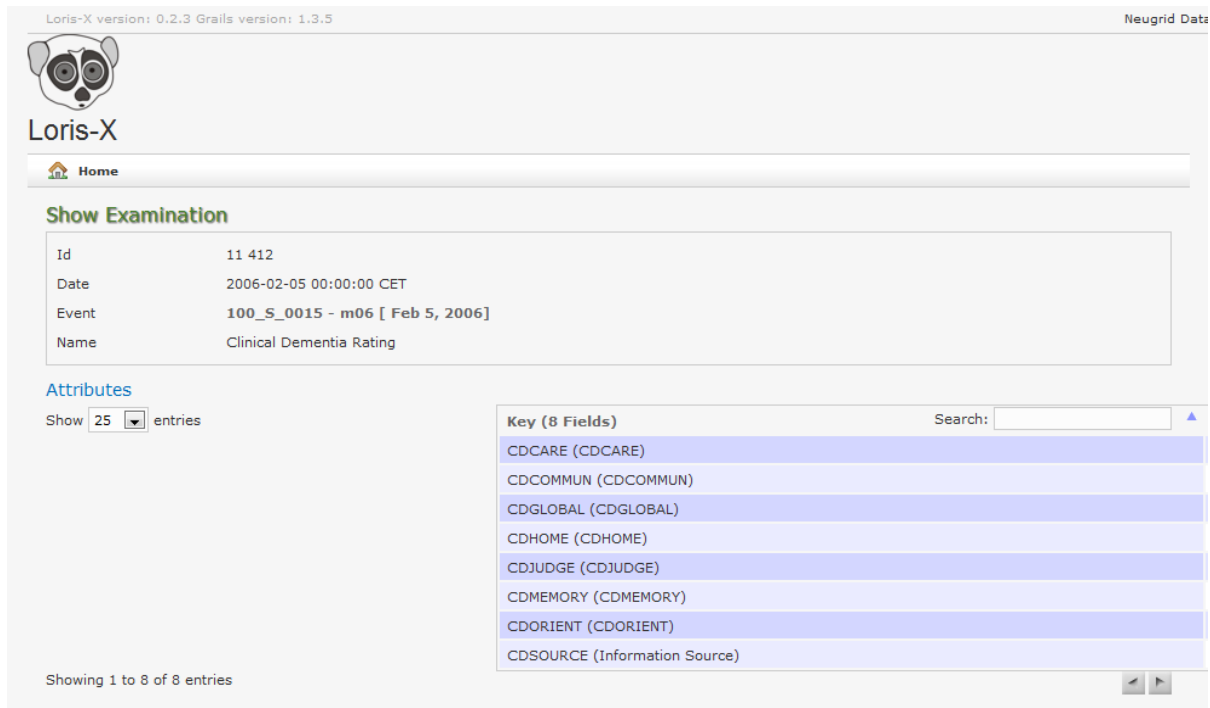



Figure 4: Event view

The subject properties seen in the subject view show all the attributes related to the subject such as demographics (date of birth, gender and years of education) and other information. These attributes will not change over the course of a study and are not specific for specific event.

In the event view all examinations connected to the event are displayed. By entering a examination of choice all the information related to that examination will be visualized (figure 5, clinical dementia rating).



Loris-X version: 0.2.3 Grails version: 1.3.5 Neugrid Data

 Loris-X

[Home](#)

Show Examination

Id	11 412
Date	2006-02-05 00:00:00 CET
Event	100_S_0015 - m06 [Feb 5, 2006]
Name	Clinical Dementia Rating

Attributes

Show entries

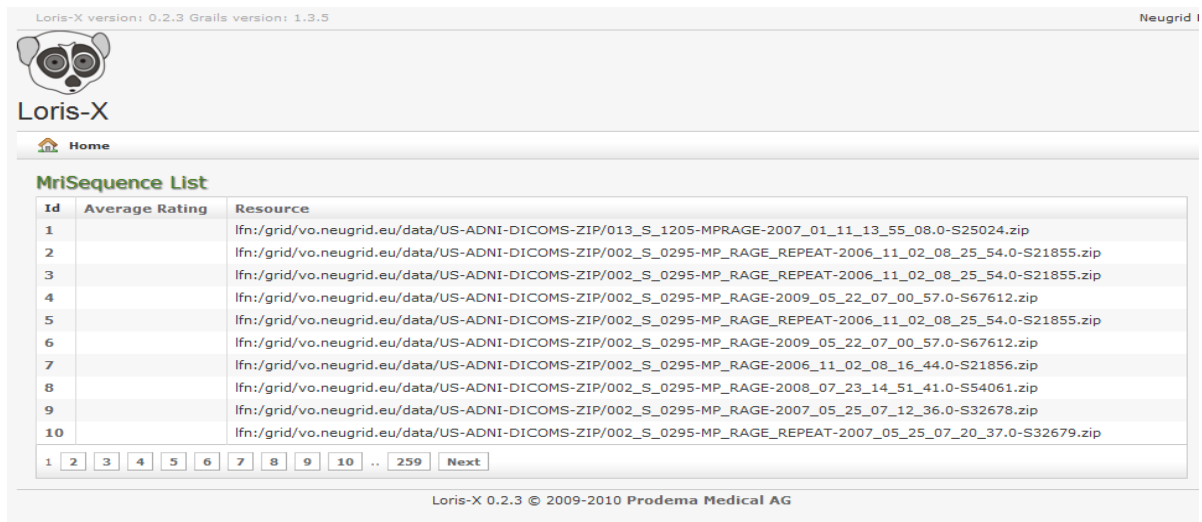
Key (8 Fields)	Search:
CDCARE (CDCARE)	<input type="text"/>
CDCOMMUN (CDCOMMUN)	<input type="text"/>
CDGLOBAL (CDGLOBAL)	<input type="text"/>
CDHOME (CDHOME)	<input type="text"/>
CDJUDGE (CDJUDGE)	<input type="text"/>
CDMEMORY (CDMEMORY)	<input type="text"/>
CDORIENT (CDORIENT)	<input type="text"/>
CDSOURCE (Information Source)	<input type="text"/>

Showing 1 to 8 of 8 entries


Figure 5: Examination view

9.2.2 Browsing by images (=Quality control interface)

From the database home menu, the second option is browsing by images.



Loris-X version: 0.2.3 Grails version: 1.3.5 Neugrid I

 Loris-X

[Home](#)

MriSequence List

Id	Average Rating	Resource
1		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/013_S_1205-MPRAGE-2007_01_11_13_55_08.0-S25024.zip
2		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE_REPEAT-2006_11_02_08_25_54.0-S21855.zip
3		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE_REPEAT-2006_11_02_08_25_54.0-S21855.zip
4		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE-2009_05_22_07_00_57.0-S67612.zip
5		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE_REPEAT-2006_11_02_08_25_54.0-S21855.zip
6		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE-2009_05_22_07_00_57.0-S67612.zip
7		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE-2006_11_02_08_16_44.0-S21856.zip
8		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE-2008_07_23_14_51_41.0-S54061.zip
9		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE-2007_05_25_07_12_36.0-S32678.zip
10		lfn:/grid/vo.neugrid.eu/data/US-ADNI-DICOMS-ZIP/002_S_0295-MP_RAGE_REPEAT-2007_05_25_07_20_37.0-S32679.zip

1 2 3 4 5 6 7 8 9 10 .. 259 Next

Loris-X 0.2.3 © 2009-2010 Prodema Medical AG

Figure 6: Browsing by image

This option is mainly in the db for quality control purposes (will be discussed in chapter 1.4). When entering this page a similar view as for browsing by subjects will be displayed (figure 6). The main difference is browsing for images, instead of browsing for subjects. By choosing an id number, the db will display the quality control interface where information related to the image can be seen and the image can be visualized in 3D, in the image viewer.

9.2.3 Querying

The database can be queried by subjects, events and images. Figure 7 displays the querying view (events). The view looks very similar for all three different types of queries. When querying the database it is important to know the relationship between subjects, events and images. Naturally, all three entities are very closely related. All images are connected to a subject and an event. All events are connected to a subject but not necessary an image. There cannot be an event or an image which is not connected to a subject. But a subject does not have to have an event or an image connected to it. Other parameters in the database such as clinical parameters and examination types are all naturally connected to a subject but not always to an event or an image. However this is not entirely true when doing a query in the database. When querying for subjects you can only query for parameters which are constant over the whole study such as date of birth and gender etc. Other parameters that might changes over time such as diagnosis and certain examinations that only take place at certain time points can only queried by events or images

Loris-X version: 0.2.3 Grails version: 1.3.5 Neugrid Data

Loris-X

Home Query Subjects Query Events Query Images

Query Events

- ▼ **1.5T MRI Scan Information**
 - MMARCHIVE - Data Archived Locally
 - MMB1BODY - B1 Calibration Body Coil Scan (Only applicable for phased array head coil on GE and Siemens systems)
 - MMB1HEAD - B1 Calibration Head Coil Scan (Only applicable for phased array head coil on GE and Siemens systems)
 - MMCONDCT - Was the scan conducted?
 - MMECHO - Straight Axial Fast or Turbo Spin Echo
 - MMLPDONE - Was a Lumbar Puncture completed prior to the MRI scan?
 - MMLPINTER - If Yes What was the interval between LP and MRI?
 - MMPHAN - In new exam
 - MMREASON - Reason why the scan was not conducted:
 - MMRMPRAGE - Repeat Straight Sagittal MPRAGE Sequence
 - MMSCOUT - Tri-Planar Scout (if available otherwise use an axial scout)
 - MMSMPRAGE - Straight Sagittal MPRAGE Sequence
 - MMTRNSFR - Was data transferred to LONI within 24 hours of scan?
- ▶ **3T MRI Scan Information**
- ▶ **Adverse Events / Hospitalization**
- ▶ **Baseline Symptoms Checklist**
- ▶ **Clinical Dementia Rating**
- ▶ **Concurrent Medication**
- ▶ **Diagnosis & Symptoms Checklist**
- ▶ **Diagnostic Summary**

Output Columns

- Drag and drop variables here

Constraints

- Drag and drop variables here

Figure 7: Querying

As can be observed in the querying view (figure 7) there are two types of fields which can be filled in, the results fields and the constraints. The result fields determine which parameters to query for and the constraints restrict the database in its search. If no constraints are chosen the db will display all the data in the database in the category queried. All the variables which the data base can be queried for are sorted under different categories, as

can be seen in figure 7. Variables chosen as outputs and constraints are dragged and dropped from the variables lists to the output and constraints fields. There are no limits on how many outputs and constraints that can be used for a query. By choosing submit, the database starts to retrieve the data. Once the database has retrieved the data, there are three options on how to download the data, as a comma separated value file (CSV), as an excel spreadsheet (Excel) or as an OpenDocument spreadsheet (ODS).

9.2.3.1 Output fields and constraints

In the output field the parameters of interest for the query is dragged and dropped from the list of variables. It is possible to query the database for several parameters at the same time, by just adding the next variable of interest. When querying for subjects, only parameters which remains constant over the whole study can be selected and queried.

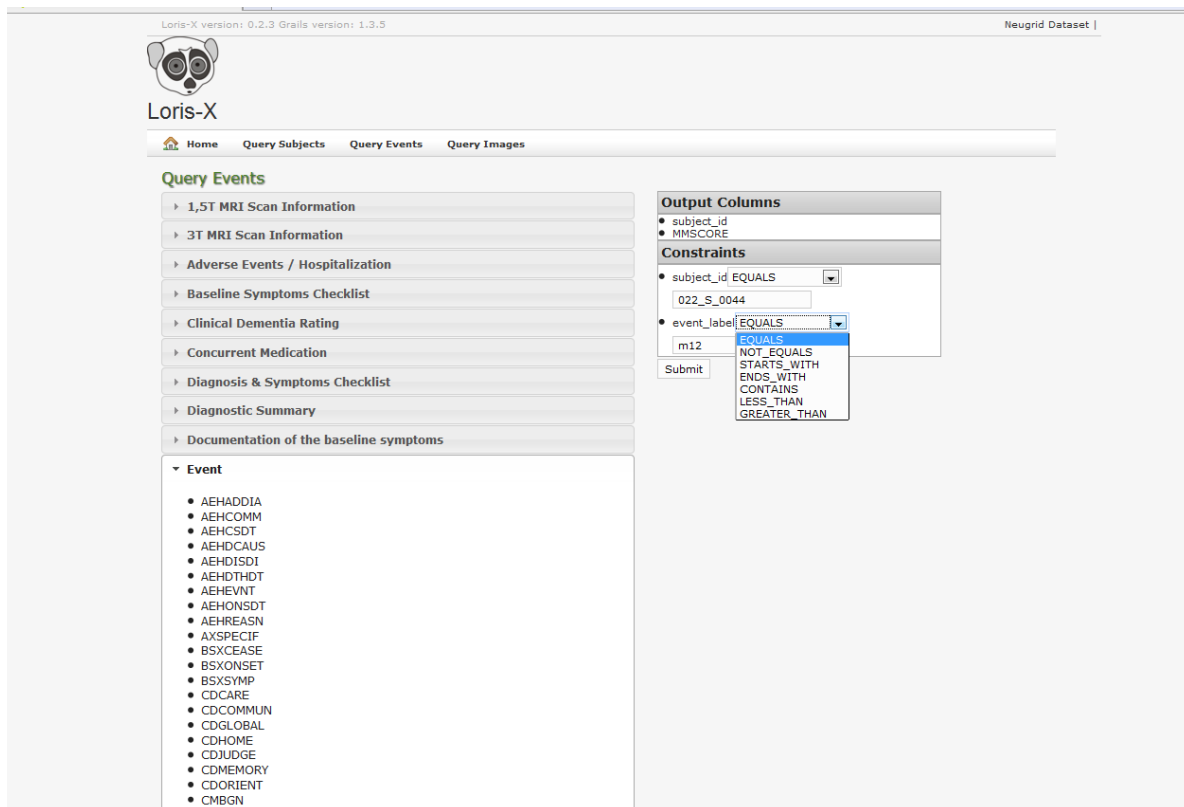


Figure 8: Query with the constraint subject id equals to 022_S_0044

Parameters which might change over time or might only exist at certain time points can only be queried by events and images.

Once all the results fields are selected, it is possible to restrict the query, using the constraints. If no constraints are added the database will retrieve all the information in the db related to the search. A query can contain several different constraints, by dragging and dropping the variables of choice to limit the search. The drop down menu for each constraint (figure 8) defines how specific the search will be regarding the constraint. The field following the drop down menu defines what the database will search for.

9.2.3.2 Example query

In this simple query example we are interested in finding out the diagnosis and the minimal score (MMSE) of a subgroup of patients which all have a MMSE less than 25 at six

month follow-up (figure 9). We are also only interested in subjects which subject id starts with 022.

The screenshot shows the Loris-X web interface. At the top, it displays 'Loris-X version: 0.2.3 Grails version: 1.3.5' and 'Neugrid Dataset |'. The main navigation bar includes 'Home', 'Query Subjects', 'Query Events', and 'Query Images'. The 'Query Events' section is active, showing a list of event categories on the left, with 'MMSE' expanded. The 'MMSE' list includes various tests like MMAREA, MMBALL, MMBALLDL, etc. On the right, the 'Output Columns' section lists 'subject_id', 'MAX_DXCURREN', 'event_label', and 'MMSCORE'. The 'Constraints' section has three filters: 'subject_id STARTS_WITH' set to '022', 'event_label EQUALS' set to 'm06', and 'MMSCORE LESS_THAN' set to '25'. A 'Submit' button is at the bottom of the constraints section.

Figure 9 Query example

4 results fields have been chosen (data to be displayed):

- Subject id
- Current diagnose
- Event label
- MMSE score

3 different constraints:

- Subject id starts with 022. The database queried for subjects starting with 022.
- Event label equal to m06 (follow-up 6 month after baseline) The database retrieving baseline data from subjects starting with 022 at time point m06
- MMSE score less than 25. The database only retrieving data from subjects starting with 022, with a MMSE score less than 25 at time point m06

9.2.4 Upload of clinical data

Clinical data can be uploaded to the db in an efficient way (figure 10). The upload is quick and easy to perform. To be able to upload data it has to be in xml-file format.



Figure 10: Clinical data upload

9.3 Viewing

The image view (figure 11) enables the viewing of images inside the data base. All the dicom header information related to the images can be found on this page. Under attributes other related information to the images, such as date of upload to database can be obtained. This is also the quality control interface of the data base which will be discussed in chapter 1.4.

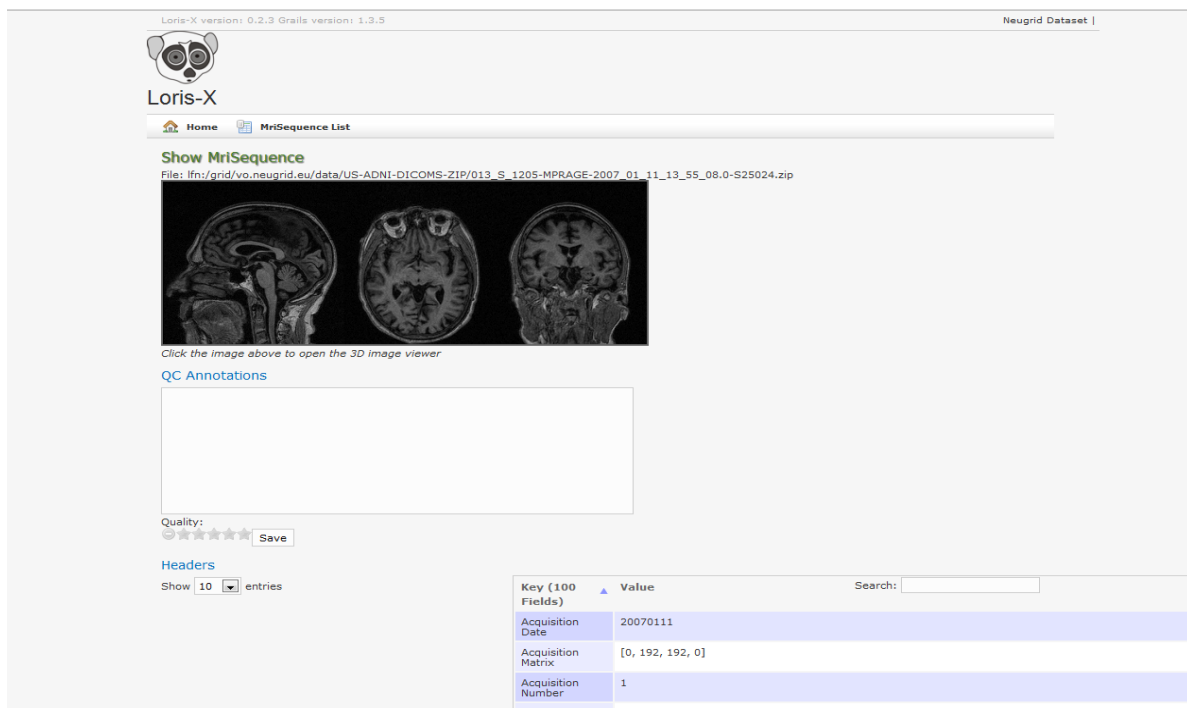


Figure 11: Image view/quality control interface

9.3.1 JIV

By clicking in the image with the left mouse button the JIV viewer will be loaded. JIV is Java software for visualization of 3D images. This viewer is used for viewing and performing the quality control.

Once the JIV window (figure 12) is loaded displayed from top to bottom, an axial, a sagittal and coronal field of view. In the bottom of the window there is a panel control area to change the contrast within the images. In the axial window z coordinate is constant in the sagittal window the x coordinate is constant and in the coronal window the y coordinate is constant. By changing the x, y and z coordinates you will change the position in the image in the following way:

- x (positive direction) left to right
- y (positive direction) posterior to anterior
- z (positive direction) inferior to superior

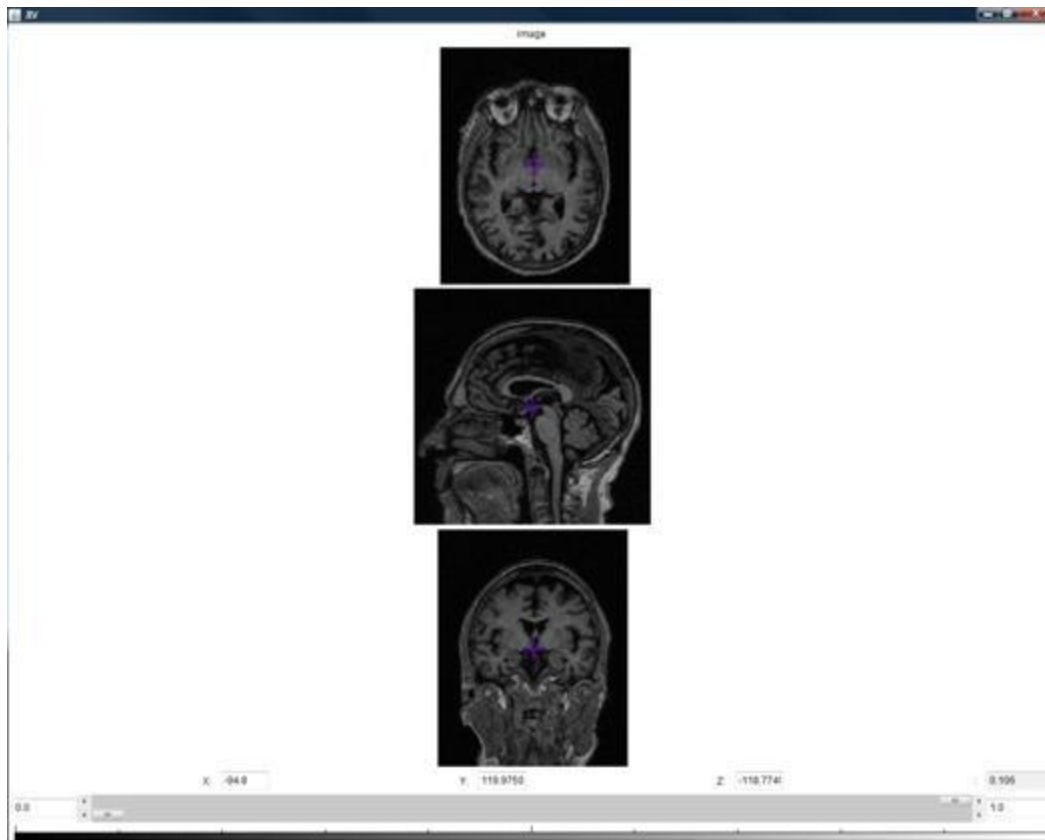


Figure 12: JIV window

9.3.2 JIV commands

The following commands can be used within the JIV window to execute certain tasks.

Left mouse button: Moves the purple cursor around in the slice plane of choice. The two other slice planes will change according to how you move the cursor within the slice plane of choice.

Middle mouse button: Hold in the middle mouse button in the slice plane of choice and move the mouse. This will move the cursor perpendicular to the slice plane of choice.

↑ and ↓ on keyboard: Place the cursor in slice plane of choice. This command will move the cursor slice by slice perpendicular to the slice plane.

Ctrl+left mouse button: By combining this command with movement of the mouse, the field of view of choice will move to preferred location on the screen

Ctrl+middle mouse button: By combining this command with movement of the mouse the field of view of choice will either zoom in or zoom out.

Double click left mouse button: This command is used to measure a distance within the image. Double click on left mouse button where you want to start the measurement and drag the cursor to end point and release button.

Double click middle mouse button: Terminates measurement.

9.3.3 Pop-up menu

By right clicking in the JIV window you will receive a pop-up menu with the following choices.

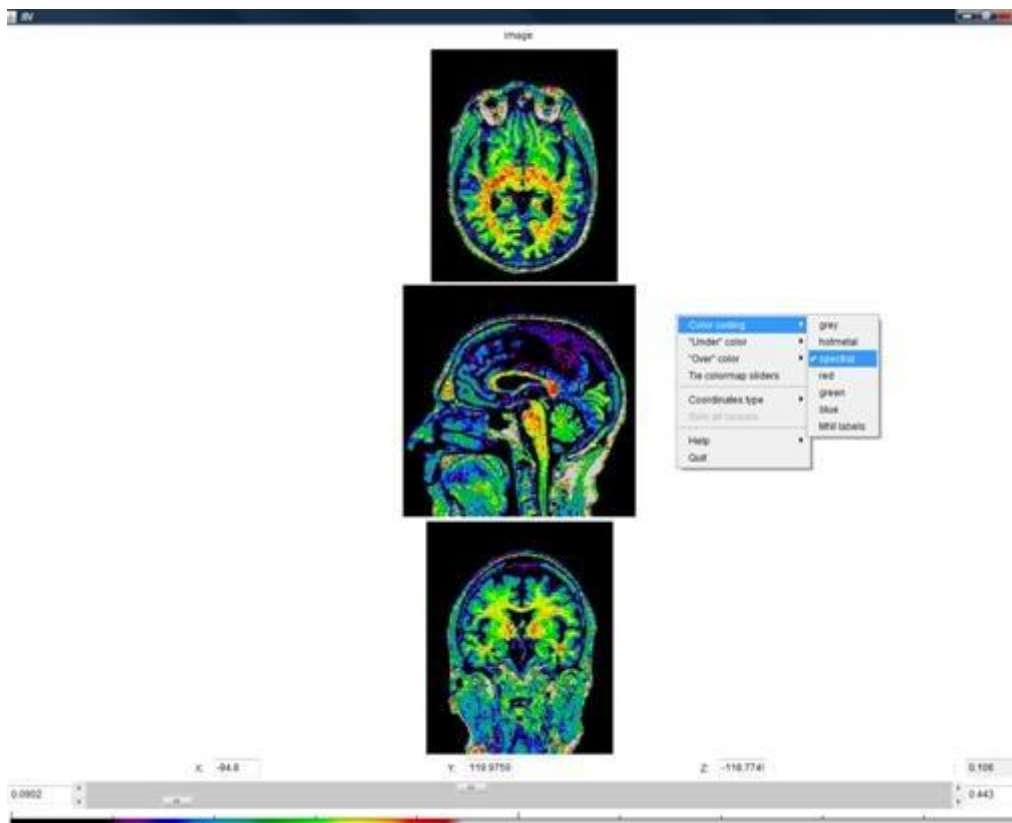


Figure 13: Color coding

Color coding: Under this category you can choose from a variety of color schemes for the image. Figure 13 show an image colored in spectral.

“under” color: changes the color of low intensity values in the image. Modified by upper slider in the control panel.

“over” color: changes the color of high intensity values in the image. Modified by lower slider in the control panel.

Tie color map sliders:

Coordinate types: In the control panel of the JIV window you find the x, y and z coordinates. Here you can change how the coordinates are displayed (world and voxel)

Help: Information related to the JIV version and copyright.

Quit: Ends the JIV session and closes the application

9.4 Quality control

Quality control is critical to any MRI study. Several factors need to be considered to achieve consistent and high quality MRI scans for image post processing. The first quality control step carried out is an automatic DICOM header and acquisition protocol verification, done before upload to the database. The second step is manual visual image control performed by trained experts within the database environment. This is performed by carefully checking the image in 3D using the JIV application. The person performing the QC is controlling for the following

- Full brain coverage
- Wrap-around artefact affecting the brain
- Motion artefacts
- Intensity inhomogeneity
- Adequate grey/white matter contrast throughout the image

Naturally, it is preferable if the QC process includes an onsite quality control (where scanning is performed) which would include the steps above, to minimize the risk of low quality images. Finally, feedback to sites and online statistics is of importance for the success of projects. Fast feedback to site will enable rescanning of subjects if necessary and prevent similar problems occurring in the future. Online QC statistics will help to monitor the project and plan future actions.

9.4.1 Quality control interface

The quality control interface is found in the image view (figure 11). The quality of the images can be rated on a six point scale, ranging from one star being very poor image quality to five stars being excellent image quality. The amount of stars corresponding to the image quality is entered and then saved by pushing the save button. There are no general guidelines on what is a good or poor image. This will have to be decided before the study starts. It mainly depends on the purpose of the study and what you want to do. If you want to use automated image analysis pipelines, there is a need for high quality images, which might not be necessary if you are only going to visual assessment.

There is also a section for adding quality control annotations. Here you will define what kinds of artefacts are observed in the image. Examples of QC annotations are found in table 1.

Table 1 QC annotations

Coverage annotations	Packet movement artefact annotations	Movement artefact annotations	Intensity artefact annotations
----------------------	--------------------------------------	-------------------------------	--------------------------------

Large AP wrap around, affecting brain	Slight movement between packets	Slight ringing artefacts	Missing slices
Medium AP wrap around, no affect on brain	Large movement between packets	Severe ringing artefacts	Reduced dynamic range due to bright artifact/pixel
Small AP wrap around, no affect on brain		Movement artefact due to eyes	Slice to slice intensity differences
Too tight LR, cutting into scalp		Movement artefact due to carotid flow	Noisy scan
Too tight LR, affecting brain			Susceptibility artefact above the ear canals
Top of scalp cut off			Susceptibility artefact due to dental work
Top of brain cut off			Sagital ghosts
Base of cerebellum cut off			