NeuroLOG ANR-06-TLOG-024
Software technologies for integration of process, data and knowledge in medical imaging

NeuroLOG Software Architecture
(draft document, January 23, 2008)

Alban Gaignard RAINBOW (I3S) alban.gaignard@i3s.unice.fr
Johan Montagnat RAINBOW (I3S) johan@i3s.unice.fr

Abstract

The purpose of this technical document is to identify the software components involved in the NeuroLOG middleware, and to precise their roles. The document describes NeuroLOG's main activities and highlight the components needed. In addition, it presents a brief survey of candidate software technologies.

1 NeuroLOG middleware users

The NeuroLOG middleware targets the secure integration of distributed data management tools (including metadata and semantic data) and image processing tools for addressing neurosciences requirements. The software will be deployed on collaborative sites. On each site, users are registered and an administrator has specific administration rights as described in the NeuroLOG Security Policy document. The two main actors in the system thus are:

- the site administrator, maintaining and deploying data and services;
- the researchers (normal users), performing local or remote operations implying local or remote data;

as illustrated in figure 1. Some of the system functionality is only accessible to the administrator.

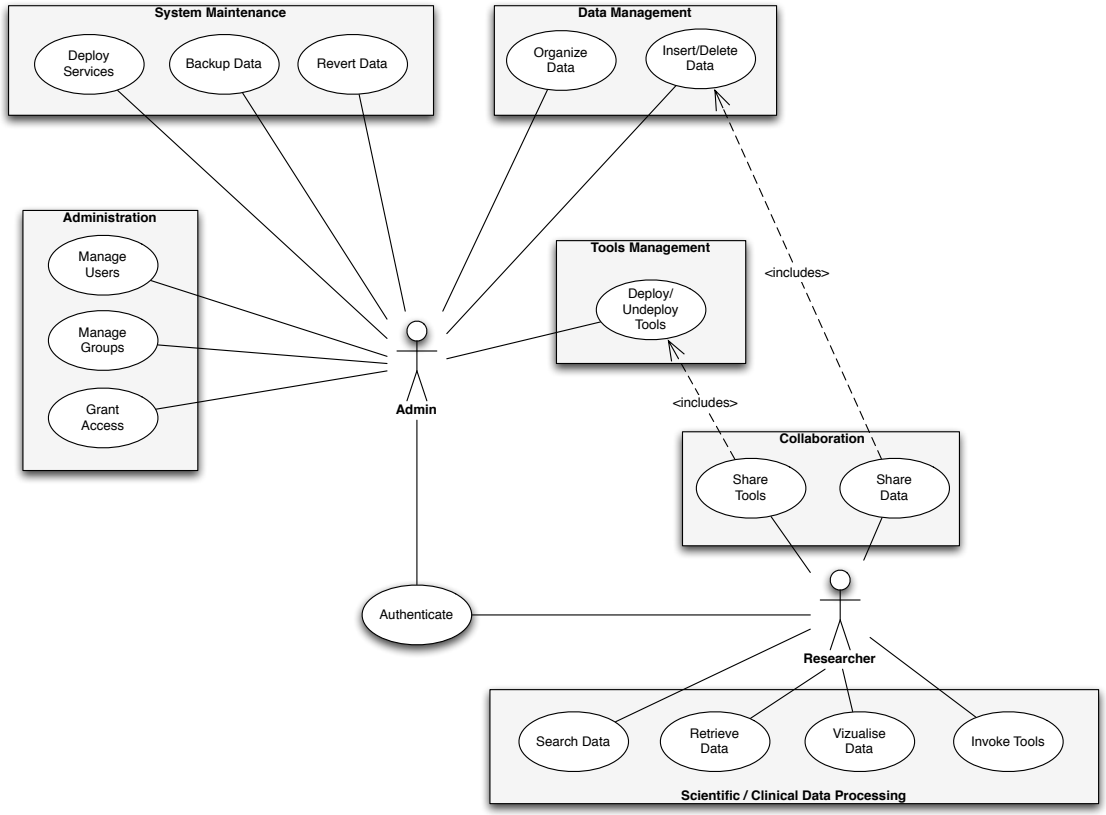

Figure 1: NeuroLOG middleware from local administrator and researcher point of view.

Figure 1 illustrates the system activities (white circles) grouped in functional boxes. All along this architecture proposal, the associated software components are described.

1.1 Administration actor

Each site collaborating in the NeuroLOG project must designate a person responsible for site administration. Note that there is no such notion as a “global” system administrator:

this makes the system more flexible, scalable and secure as the administration load is distributed on sites and the administration privileges are confined to his or her site resources. The system access right policy is thus a collection of site policies.

1.2 Research actor

Normal users have controlled access to collaborative tools for managing and processing data. The middleware provides shared data and shared algorithms functionality. It is up to the users to determine what data sets and what processing tools can be accessible to other sites. Data is anonymized and encrypted while transferred outside a site for security reasons. All communications between sites need to be authenticated and secured as well.

2 Architecture overview

This section presents a general and simple view of the middleware, decomposed in functional subsystems. Figure 2 shows each subsystem and its dependencies. All subsystems depend on the *Global Registry* which aim at ensuring the registration of all sites and the uniqueness of site identifiers. This registry will also be involved in discovery processes, for instance user groups spreading over sites, remote processing tools discovery, etc. The *Administration* subsystem is responsible for users and groups management and authentication services. The *Data* subsystem performs file operations, locally or remotely, such as file transfer, storage, browsing and other ones specific to grid storage resources. This subsystem needs to know about the existence of other partner sites (through the *Global Registry*) to perform remote operations. The *MetaData* subsystem permits local users to access metadata associated to the data sets. Those data sets are physically stored in files (either on grid resources or local resources) and are accessed through the *Data* subsystem. The *Semantic* subsystem relies on the *MetaData* in order to populate semantic data. Finally *Processing Tools & Workflows* rely on *MetaData* and *Data* subsystems to construct jobs and to transfer required data to computing units.

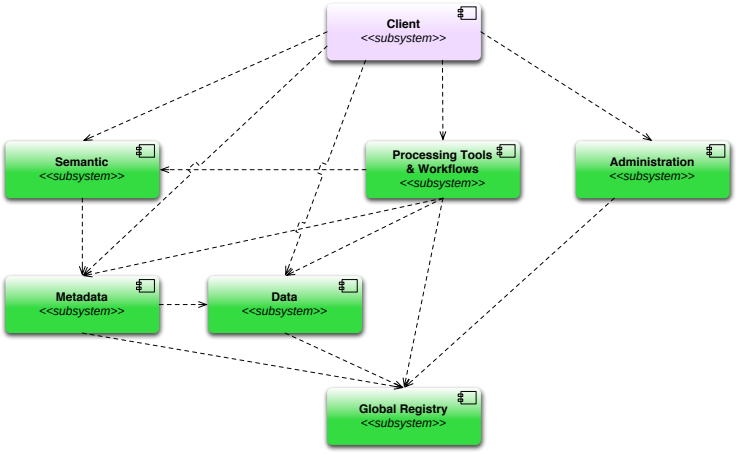

Figure 2: Software subsystems

3 System deployment

The system components instantiation depends on the platform deployment. Figure 3 shows two sites, the global registry presented above which should be deployed in one of the partner sites, and the grid computing and storage resources. The *Semantic* service (in red) only needs to be deployed in one site (*Site 1* in this example). All other services (in green) are deployed within a NeuroLOG server (one per site). Those services refer to the subsystems shown in figure 2. The tree main layers are:

- **Client layer**: the system is accessed through a lightweight web-based client or a heavyweight client application. DICOM manipulation features which should be included within one of the clients are not represented here.
- **Server layer**: the core of the system, and the service provider for each collaborating site. In order to access to resources held by other partners, each NeuroLOG server should communicate with their peers.
- **Resource layer**: within a partner site, it is possible to manage local computing resources or local storage resources.

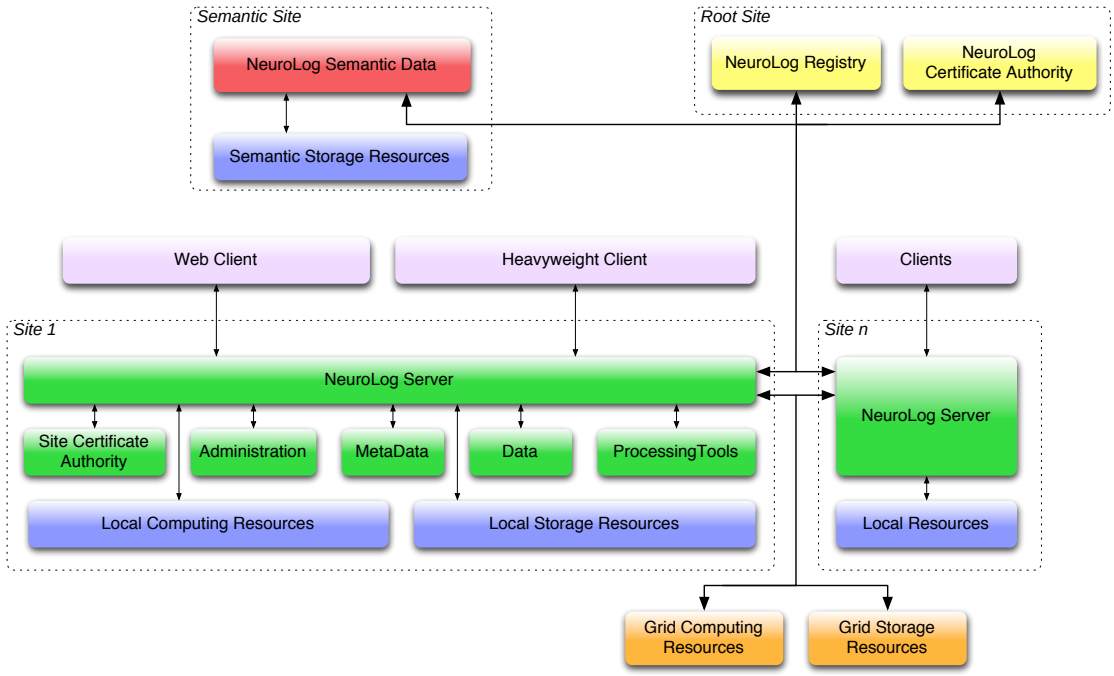

Figure 3: Platform deployment

4 Detailed architecture

This section describes the internal parts of the subsystems presented in figure 2. Those subsystems do not focus on non-functional properties like security. Thus we will spread the discussion of this topic within the description of each subsystem.

4.1 Users and groups management

Development participants: **Visioscopie**, I3S. Figure 4.

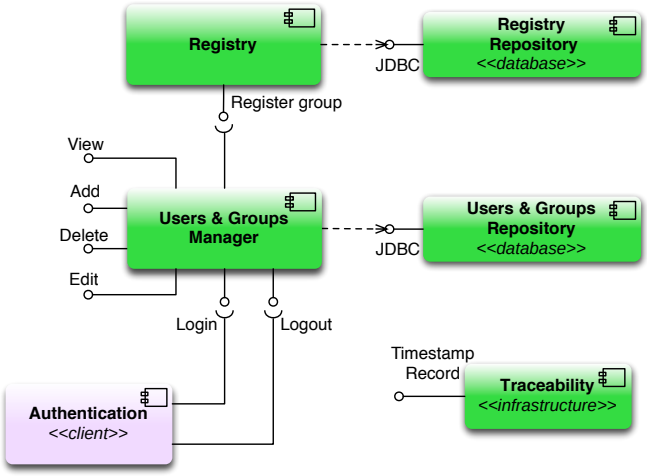

Figure 4: Users and groups management

To manage users and groups of users, the system must store their description in a repository. We introduce two components involved in this activity:

- At the root level, the global **Registry** component register sites and group names in its database. At this point of description we do not constrain the choice of a technical implementation, but for interoperability purposes, a JDBC[1] interface is required.
- At the site level, the **Site Repository** database stores the description of users and groups. The **User & Group Manager** component is a gateway to query the user and group repositories, to add, remove or edit elements. Moreover this component provides login and logout facilities. Security is an important part in this component as shown in the use cases: a local administrator is supposed to grant access to users or groups to data, meta-data, semantic information, or processing elements.

The *Traceability* component refers to the capability for the system to provide time-stamped logs as specified in the deliverable L3[2] in section 4.3.4, page 34. Each subsystem is supposed to associate time-stamp records to system elements and we assume that an administrator ask the system for admin, data, or process centric reports.

[1] Java Database Connectivity
[2] *Specification of the NeuroLOG architecture components*

4.2 Data management

Development participants: **IRISA**, I3S. Figure 5

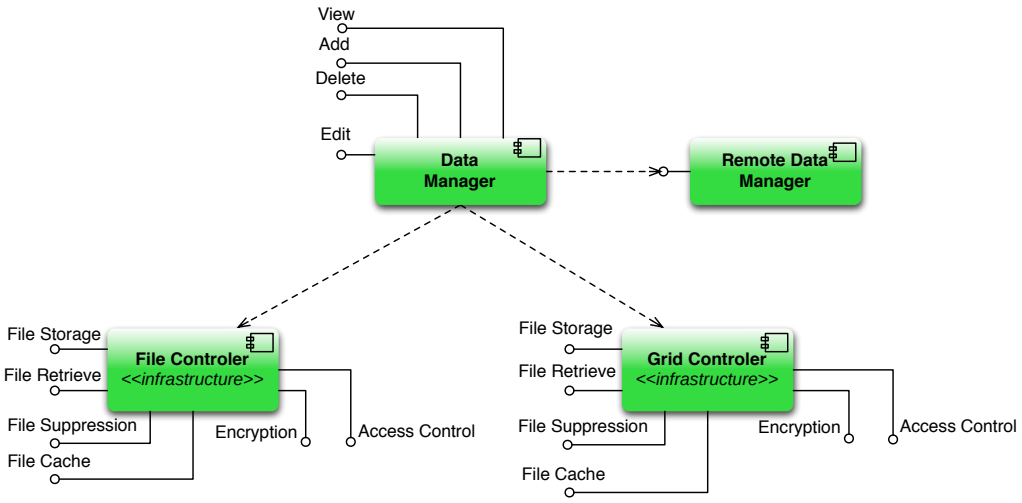

Figure 5: Data management

The Data subsystem is responsible for browsing, retrieving and storing local or remote data, either on NeuroLOG partner sites or on the grid. The *File Controller* component provides storage, retrieval and removal primitives for local files. We assume that this component is potentially supported by a distributed file system. The *Grid Controller* realizes the interface with the grid middleware and provides the same primitives as the *File Controller*. In addition, a cache primitive is provided to limit data transfer in case of multiples requests for files.

In order to securely store and transport data, and to prevent non authorized accesses to files, both controllers provide security primitives for file encryption and file access control and require authentication capabilities provided by *Administration* subsystems. As a consequence, the *Data Manager* component depends on the *Administration* subsystem.

4.3 Metadata

Development participants: **BO**, IRISA. Figure 6

The metadata is composed of two databases:

1. site-specific metadata structured using a site-specific schema; and
2. metadata stored using a NeuroLOG common schema.

The NeuroLOG middleware can directly read and write metadata stored in the common schema repository through the *Metadata Manager* component. Metadata store in the site-specific repository is written by a site specific tool. A mediator is proposed to query this metadata through *Data Federator*. *Data Federator* can thus access both site-specific and common-schema metadata. In addition it provides access to remote sites metadata.

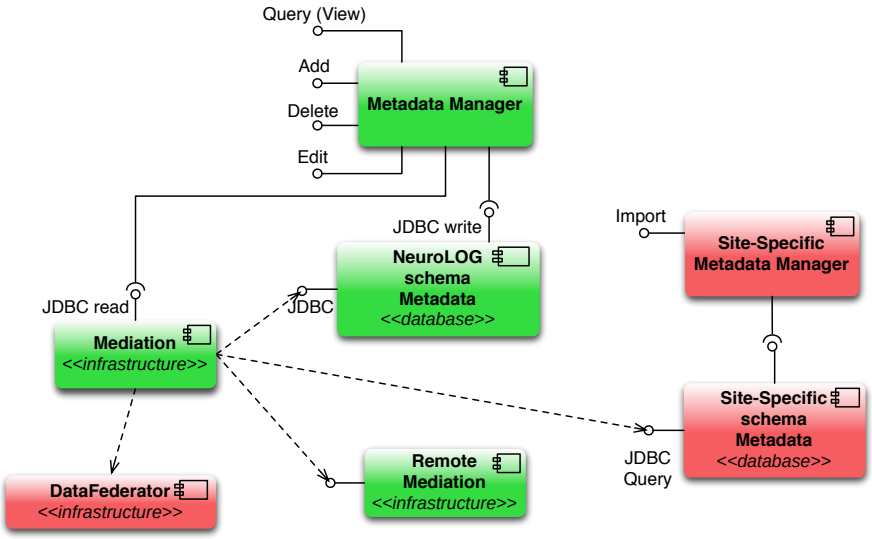

Figure 6: Data management

4.4 Processing tools and workflows

Development participants: **I3S**, LRI. Figure 7

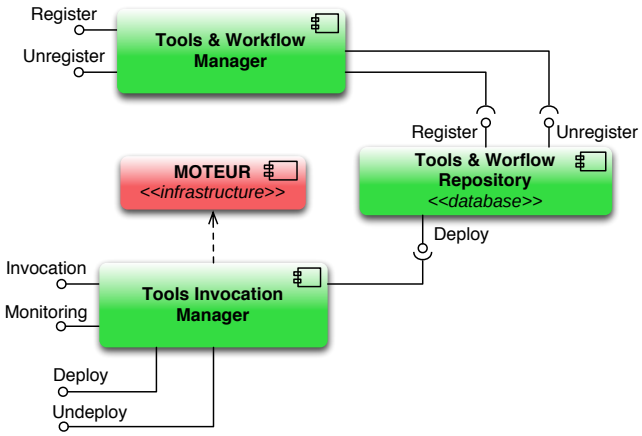

Figure 7: Tools and workflow management

Another administration task is, initiated by a researcher, to publish or unpublish processing tools or workflows. It is possible through the *Register* and *Unregister* primitives of the *Processing Tools & Workflow Manager* component. A precise description of tools and workflows (eventually including either binaries or workflow description files) is stored in a repository. This section of the document does not focus on technical implementation of repositories.

One of the main research activities pointed out in the use cases section is the invocation of tools. Prior to that, tools must be deployed (potentially undeployed) over execution

hosts. The *Processing Tools Invocation Manager* encapsulates *MOTEUR* software and provides tool invocation and monitoring primitives aimed at this activity. This component relies on the *Grid Controller* for job submission over the grid.

4.5 Semantic information

Development participants: **LaRIA**, IRISA. Figure 8

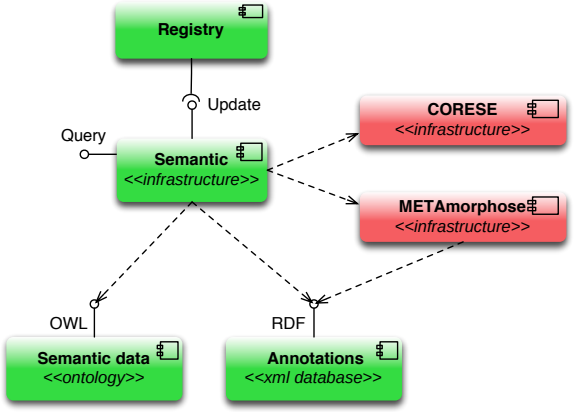

Figure 8: Semantic information management

The *Semantic* component aims at querying ontological information (through the *CORESE* software) and thus retrieving data with rich queries. The semantic annotations are stored in a RDF repository generated from the *Metadata* subsystem by the *METAmorphoses* software. It implies dependencies with *Data* component for retrieval, and *Metadata* component for queries. The ontology in stored in an OWL database.

4.6 Clients

Development participants: **Visioscopie**, I3S, IRISA.

Two clients will be developed to communicate with the NeuroLOG servers: a heavy client will offer full functionality while a web-based lightweight client will provide access to a limited subset of functionality: query interface and limited visualization capabilities.

5 Technologies survey

5.1 Communications between distributed components

The client interface and server code will be written in java. The client will be interfaced to a C# visualizer. Internal communications between the java client and the C# visualizer can be performed through local network socket or shared memory. The components communication channels are illustrated in figure 9: the web client interfaces to the webapp container through secured HTTP while the heavyweight application client uses RMI.

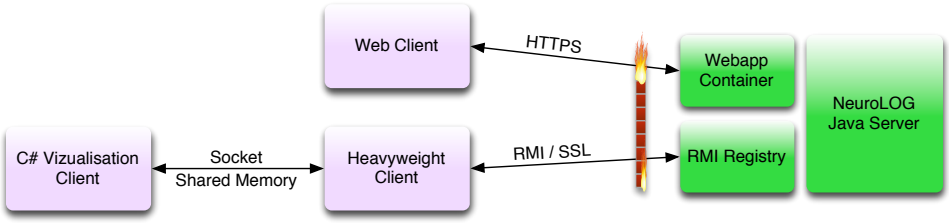

Figure 9: Communication scheme between clients and server

Remote communications between the server and the client require authentication and security. Different technologies have been considered:

- Java RMI
- CORBA
- Web Services and Grid Services (WSRF)
- ICE[3]

The Java SE Security (JSSE[4]) provides an X509- and TLS-based authentication and secured communication scheme. Clients and servers are identified by cryptographically signed key pairs. X509 certificate chains can be validated to identify users owning Certificate Authorities signed certificates. The Java RMI protocol (JRMP) can be initiated on top of secured sockets. The combination of RMI and JSSE provides authenticated and secured remote procedure calls, compatible with standard certificates such as the EGEE user certificates. In addition, the JRMP handles objects serialization/deserialization and code invocation transparently, without requiring any additional protocol definition. It is an interesting complete and lightweight solution readily available inside Java (since version 5). It is to be noted that file transfer will require a specific protocol (*e.g. GridFTP*): for large volumes of data, JRMP is not reliable enough. The data management components will therefore need to expose a specific interface for file uploads and downloads.

Internally to one site, all server components should be deployed on a single host to avoid network exchanges. The use of enterprise java beans components (as opposed to simple java objects) is being investigated for writing and deploying the various software components on the server side. More lightweight solutions are also being considered such as the use of the hibernate library for persistent objects implementation. 8

[3] http://www.zeroc.com
[4] http://java.sun.com/javase/technologies/security

5.2 Users authentication

Operational security requires to assign:

1. signed certificates to users, to control user access right to resources; and
2. public/private key pairs to clients and servers, to control trust between them and correct handshaking on communication establishment.

The server certificate will be generated at server installation time and managed by the NeuroLOG middleware on behalf of the site administrator. The client certificate will be generated by the site administrator at client creation time and managed on the client host by the NeuroLOG middleware.

Users wishing to access the EGEE grid resources will need EGEE certificates delivered by the CNRS Certificate Authority. For access to other NeuroLOG resources, EGEE certificates could be used as well. The pros is to delegate certificate generation to an existing and reliable CA. The cons is to depend on an external entity: no user can access the system without prior registration to the EGEE CA. Alternatively, the NeuroLOG project can set up its own CA and generate certificates to its users. Those certificates will only be recognized by the NeuroLOG middleware, not by the EGEE middleware, hence both certificates should be manipulated if access to EGEE resources is needed. The pros is the independence of the NeuroLOG CA from EGEE. The cons is the additional load on the project administrators delivering the certificates and the additional complexity of the middleware (double certificates and CA management).

A NeuroLOG CA deployment scenario is illustrated in figure 10. To avoid overload of the root CA, each site deploys a sub-CA that is recognized by the root CA and the other participating sites. As a consequence, each site can directly sign the certificates for its server and its users.

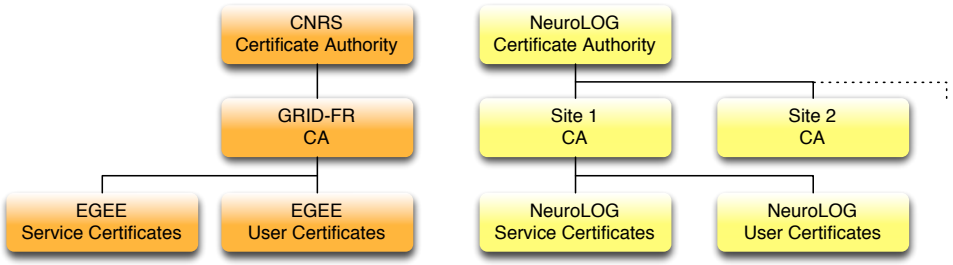

Figure 10: Grid and NeuroLOG Certificate Authority hierarchies

At a higher level, the users will be authenticated and mapped to their certificate(s) on the client side after login/password or CPS-based authentication. The certificates and the internal complexity will be hidden.

5.3 IDE

The SUN NetBeans IDE appears to be a good compromise, lighter and more flexible than eclipse.

6 Two practical use cases

This section presents two practical use cases in order two extract and delimit functional of NeuroLOG software components. Pink blocs refer to the client application side whereas green ones refer to the server application side. The first one explains how a user can insert new medical images within the system, the second one explains how data can be processed through tool invocation.

6.1 DICOM Image insertion

The data registration workflow from a DICOM media is illustrated in figure 11. The user starts his NeuroLOG client and authenticates. This step ensures that the user can access to the system (or some sub parts of the system) as a duly identified actor. From this step an authenticated session is started between the client and the NeuroLOG service provider or every messages exchanged are signed.

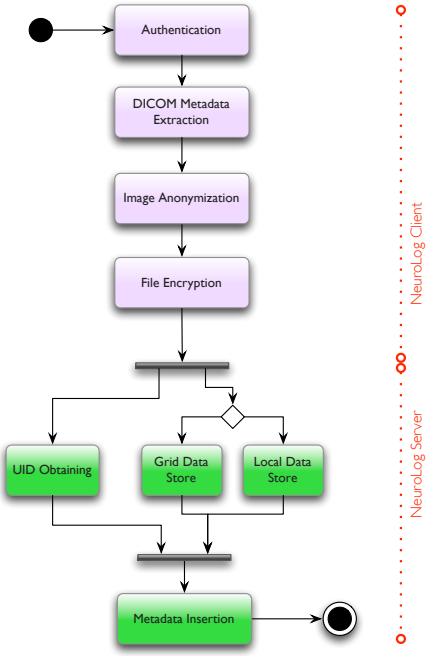

Figure 11: Activities involved in DICOM image insertion

The second step extracts meta-data contained in the DICOM image(s). Sensitive meta-data (hidden from anonymization process) will be encrypted and separated from other meta-data. All other meta-data will be extracted and stored in a structured document (an XML file for instance).

The next step realizes the anonymization of the input DICOM data following the DICOM standard. The resulting file cannot be re-identified on its own.

For security constraints data (raw files and meta-data) that is submitted to the system (i.e. travel over the network between the client and the server) is encrypted. After the encryption step, the user hosts on the client side:

- encrypted non-sensitive meta-data

- encrypted sensitive (nominative) meta-data
- a set of encrypted DICOM files

This package of data can now be securely transmitted over the network to the server responsible for coherent data insertion.

The NeuroLOG server receives a data insertion request (refers to the "add" interface of the data manager in the figure 5) and starts a transaction. It consists in two independent activities, obtaining a unique identifier for data to insert, and realizing effectively the storage (supported by the grid or file controller of the figure 5, it depends on the kind of data insertion request). Then if everything is successful, the data manager updates the meta-data with location and file access information and finally the transaction is committed. The server then insert meta-data in the system in a way that prevent non clinical users from accessing sensitive data. If the meta-data insertion fails, the previous transaction is reverted with a roll-back mechanism, otherwise, the server notify the client of data acceptance.

A lightweight version of this use case can be deployed locally for the needs of testing and data registration:

- Skip user authentication and encryption stages (local execution only)*
- Define an XML schema for DICOM metadata extraction
- Map DICOM metadata to the site specific data schema (the NeuroLOG common schema by default)
- Add the association between file name and file UID in the metadata
- Use JDBC driver to perform metadata registration in the local database.

6.2 Tool invocation

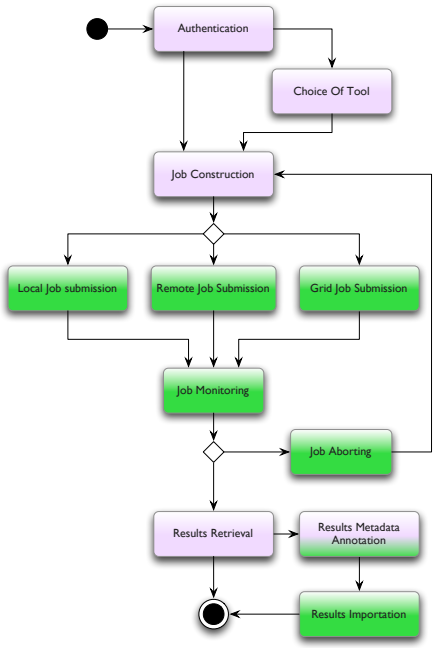

Figure 12: Activities involved in tool invocation