# A Grid Enabled Medical Image Database

**A.L. Rowland**[*], T. Hartkens[†], M. Burns[‡], J.V. Hajnal[*], D. Rueckert[‡], D.L.G. Hill[†]

Imaging Sciences Centre, Imperial College London [*]
Department of Computing, Imperial College London [‡]
Division of Imaging Sciences, King's College London[†]

## Abstract

Medical image processing involves a large amount of data that can be difficult to store and manage. This project uses an automated process to extract clinical and technical information from image file headers which is used to construct a database of medical images. A web based search and retrieval interface is used to present the data to the user with on-line access to images via a choice of viewing tools. A grid-enabled image processing portal is provided with access to a suite of image processing algorithms implemented as grid services which can then be seamlessly invoked from the web interface.

## Medical Image Metadata

Digital Imaging and Communication in Medicine (DICOM) is the industry standard format for medical imaging. Each individual 2D image slice has its own distinct file complete with extensive file header containing a wide variety of information including:

- Patient details and demographics
- Study specific details (e.g. date, time, duration, scanning protocol)
- Representational information such as byte ordering and transfer syntax
- Scanner details and image acquisition parameters

This metadata is of critical importance to researchers and clinicians alike. The standardised format of DICOM lends itself particularly well to parsing by scripting tools such as Perl. The parsed data from each field can then be used to populate a relational database where it can be later queried.

## Data importation

There are several steps involved to obtain the image data in a usable form. These include:

- Data transfer – exporting image data from a scanner to a host on the local network that that implements the DICOM *storage class provider* interface.

- Parsing of DICOM files and insertion of this metadata into the relational database together with the location of the related image files.

- Reconstruction – sorting and combining the many individual 2D image slices into one or more volume or temporal data sets before conversion to a preferred image file format.

## Interface

The web interface provides a user friendly front end to the image database[1]. Users can search for subjects matching various criteria such as demographics, scanning protocol and clinical diagnosis.

The user may then 'drill down' into the appropriate study and sequence to reveal more detailed information. A variety of technical and clinical details are displayed together with any related images, which may be viewed directly using an integrated viewing tool running as a Java applet.

Other facilities have been added to re-export data to recordable media and to do simple tasks such as file format conversions, slice extraction from image volumes and creation of video sequences from time series datasets.

Images acquired directly from the scanner often require a degree of modification to be of any value. Procedures such as image segmentation (labelling tissues in the image) and image registration (alignment of one image with another to enable comparisons) are particularly common.

There are many tools that can perform these tasks from a number of different providers, each with their own requirements and their own way of doing things. Additionally certain algorithms are quite computationally intensive placing a heavy burden on the hardware. For example non-rigid registration (free-form deformation) of a typical MRI image volume may take around six hours to complete on a modern Pentium 4 PC.

Image processing tools are therefore an essential adjunct to an image database, however clearly do not belong as an integral part of the system.

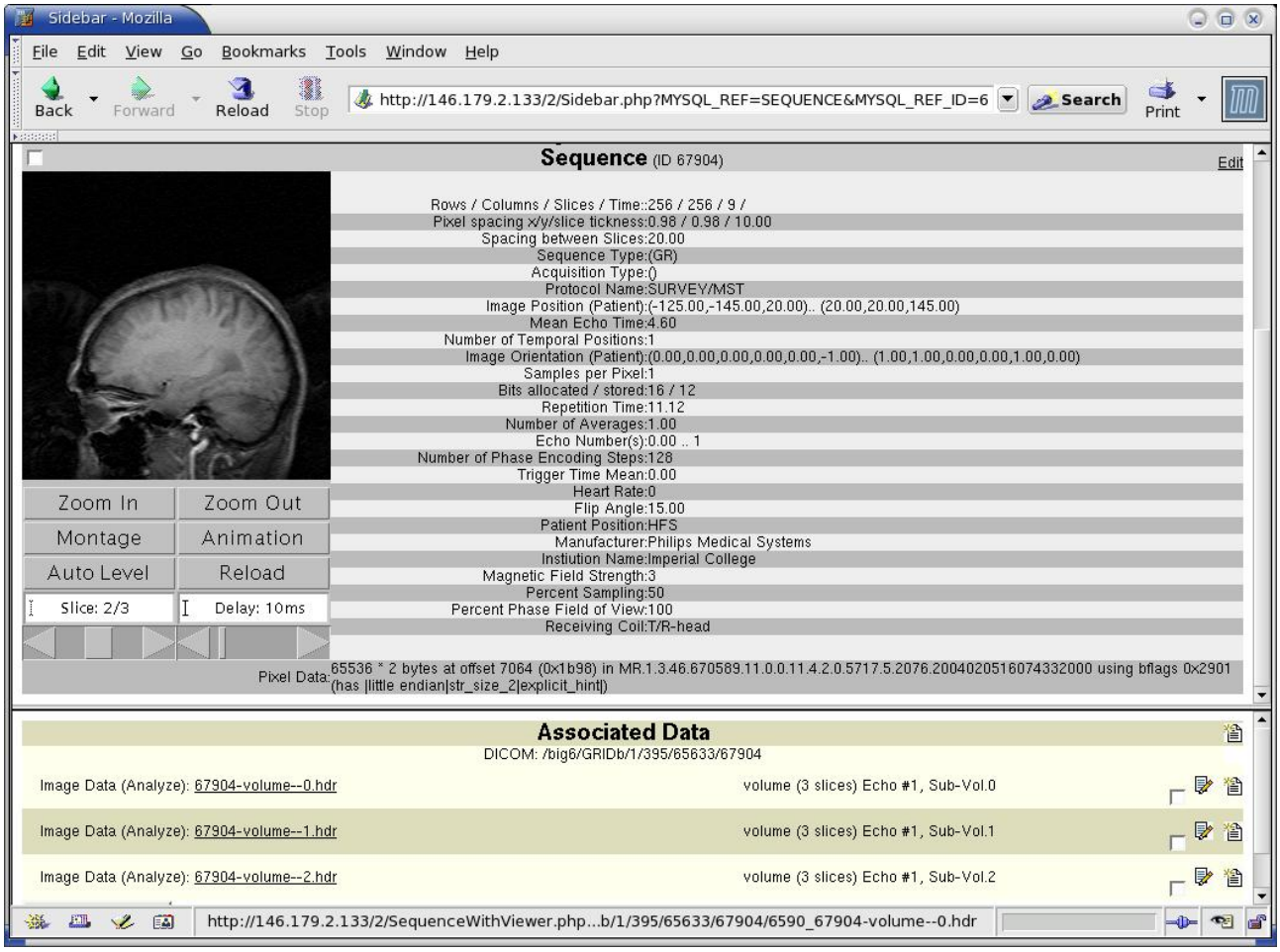


Fig1. Screenshot – Image metadata and integrated viewer

**Grid Service Integration**

The design philosophy has therefore been to represent different image processing algorithms as *grid services* which may exist elsewhere on the grid often associated with their own dedicated, high specification hardware. From an abstract viewpoint, each service can be seen as taking a certain number of inputs and producing a certain number of outputs. This led to the development of the Medical Imaging Component Language[2] (MICL) – a set of core requirements which allows a diverse number of algorithms from different providers to conform to a more generic representation. This allows the services to appear as modular components that can be combined in any order irrespective of their source or physical location.

Grid services are invoked indirectly via the web interface using Java servlet technology. The Globus CoG toolkit includes an API that makes tools such as Grid FTP and x509 credential management easy to use and which integrates directly with the code used to generate the web page.

**Image processing and workflows**

Image processing, within the context of medical imaging, is rarely a one step procedure. More commonly a series of algorithms are applied to produce an image processing pipeline where the output of one step in the chain is fed in as one of the inputs of a subsequent step.

A simple workflow language was developed to describe this pipeline, relating the values of the inputs and outputs and the component grid services to which they refer. In effect a workflow becomes a kind of wiring diagram, specifying how stages link together and to which particular component services they refer.

The idea of being able to create a workflow template and store it for reuse was of critical importance. Certain workflows are particularly common and being able to retrieve a stored copy would save considerable time as well as reducing the chances of human error.

Workflow management can be broken down into creation, storage, retrieval and instantiation

Creation**:** Workflow templates, which we refer to as *abstract workflows* are constructed via an HTML interface which queries a Registry Service for available grid services. Component stages of a workflow can then be ordered and their inputs and outputs linked together.

Storage: Although the workflows are used internally as Java objects, for storage they are represented as XML and stored in a native XML database (eXist[4]). This has the advantage of allowing workflows to be queried using Xpath/Xquery where metadata such as the services referenced or various options or parameters can be searched.

Retrieval: Stored workflows can be selected from a summarised list, or be browsed via an associated style sheet. The Java language has an excellent API for handling XML data, allowing any chosen workflow retrieved from the database to be converted directly to a Java object which can then be used for instantiation and job submission. Commonly used abstract workflows may then be simply picked 'off the shelf' with their inputs and outputs already wired up correctly. All that is required prior to submission is to instantiate the workflow against a specific set of input files.

Instantiation: Abstract workflows such as those created and stored in the steps above, are effectively templates. Workflow stages are linked using references and logical file names. Before the workflow can be submitted for execution, these logical placeholders must be instantiated with the actual files required. This final step is performed just prior to submission and increases the efficiency of the workflow by permitting different instances of the same algorithm to be placed in the same workflow stage. This will have the effect of informing the Workflow Service that each different instance can be run concurrently.

**Typical usage**

A typical usage scenario would be as follows:

1. Initial database query
2. Selection of images for the workbench
3. Workflow construction or retrieval of a stored workflow from the database
4. Workflow instantiation using images from the workbench
5. Submission of a workflow for execution using secure encrypted methods for data transfer
6. Receive notification of completion
7. Choose which files to re-insert into database
8. View results

From the sequence above:

- Stages 1, 2 and 8 require SQL database querying and dereferencing of file locations.

- Stage 3 requires XML database querying using Xpath and XML-DB to retrieve workflow related information

- Stages 3, 4, 5 and 6 involve calls to a minimum of three different grid services.

However, the user does not deal directly with databases or grid services - the interface hides this complexity from view.

The user interacts with the *workflow* which they have either constructed themselves or have retrieved from a stored collection. Effectively the workflow acts as a kind of black box – it contains the calls to the grid services, yet these remain invisible to the user. Once the user has configured the inputs to the workflow, everything else is handled internally.

This abstraction is probably best explained by analysing the sequence of events that occur and what the user sees from his/her perspective (fig 2)

For a more detailed description of the how a workflow is executed after submission, the reader is referred to [2] as this is, in itself, a complex process involving delegations to other services over several layers.

| Task performed | User action required |
|---|---|
| Search the database for images matching certain criteria | Type in some keywords<br>Click *search* |
| Add the chosen images to the workbench | Tick the check box for each image |
| Choose a workflow template from the XML database | A list of stored workflows is retrieved automatically<br>Select an entry from a supplied drop-list |
| Specify which inputs to use for required template parameters | Select options from the supplied drop lists (type safe) |
| Choose a name for any output files | Type a name in the supplied text box |
| Invoke the Workflow Service on a remote network | Click the submit button |
| [ Workflow Executing ] | |
| Receive notification of completion and access the Reinsertion Service to review the results | Open an e-mail and click on the supplied web link |
| View the images and other associated intermediary files | All results are hyperlinked to the file system<br>Image files launch an integrated applet viewer. |
| Choose which files to reinsert into the database | Tick the appropriate check boxes |
| Copy the selected files to the appropriate location in the file system and add a corresponding entry to the relational database. | Click the reinsert button.<br>File locations are determined automatically by recursive lookup of input files in the workflow |

Fig 2. Typical event sequence before, during and after submission

## Conclusion

This project facilitates access to medical image data and provides easy access to highly specialised, remote image processing services. By using a high level web interface, the complexity of the system is hidden from the user who, at no point, needs to know anything about the location of their images or where they are being processed. Grid technology has the potential to greatly improve the speed and user-friendliness of image processing tasks, opening up many, more advanced, but significantly more demanding, techniques which are currently reserved only for special situations. Tasks that previously took hours to prepare and execute can now be achieved in a few mouse clicks.

## Acknowledgements

The group would like to acknowledge the assistance of the UK E-Science program for their assistance in funding the *Information eXtraction from Images* (IXI) project of which this work forms a part. We would also like to thank GlaxoSmithKline, Philips Medical Systems and the Dunhill Charitable Trust for their additional support and funding.

## References

[1] T. Hartkens *The Guys Research Image Database* http://www-ipg.umds.ac.uk/t.hartkens/project /gridb/index.html

[2] M. Burns, A.L. Rowland, J.V. Hajnal, D. Rueckert, D.L.G. Hill *A Grid Infrastructure for Image Segmentation and Registration* UK e-Science All Hands Meeting 2004.

[3] A.L. Rowland, T. Hartkens, M. Burns, J.V. Hajnal, D. Rueckert, D.L.G. Hill *Information eXtraction from Images (IXI): Image Processing Workflows Using A Grid Enabled Image Database* DiDaMIC Workshop 2004

[4] Wolfgang Meier. *eXist: An Open Source Native XML Database* In Akmal B. Chaudri, Mario Jeckle, Erhard Rahm, Rainer Unland (Eds.): *Web, Web-Services, and Database Systems* NODe 2002 Web and Database Related Workshops, Erfurt, Germany, October 2002. Springer LNCS Series, 2593.