[osmia rufus]

OSMIA

Title:	Open Source Medical Image Analysis
Acronym:	OSMIA
Date:	16th October 2001
Project Number:	IST-2001-34512
Deliverable:	D3.1 Working Specification of OSMIA Interface
Version:	1.0
Author(s)	Tony Lacey, *University of Manchester* Ian Poole, *Voxar Ltd.*

Contents

1 Nomenclature

Abbreviation	Full Meaning
DICOM	Digital Image and Coomunications in Medicine
HTTP	HyperText Transfer Protocol
PnV	PlugNView 3D
TPS	Third-Party Software

2 Introduction

This document reports the current *working* state of the OSMIA interface system to the TINA open source image analysis environment [`http://www.tina-vision.net`]. As a working specification the contents are subject to change and update as the protoype system is developed. The specification will be superceeded by future releases (with higher version numbers). **All working specifications will be superceeded by the public release specification in develiverable D3.4.**

The purpose of the OSMIA interface is to enable commercial medical image analysis/visualisation software to rapidly access algorithms within the TINA open source image analysis environment, with minimal programming effort.

TINA provides a set of libraries which can be linked to other software. The obvious mechanism to combine TINA with other software, such as PnV, is therefore to generate dynamically loadable objects (dll's in MS Windows) which can be linked when needed. However, there are several drawbacks with this tightly coupled approach;

- The robustness and security of the commercial software could be compromised by the TINA objects.
- Extra development of the host software is required to enable integration of loadable objects.
- The technologies used to construct dynamic objects are often platform specific necessitating code differences.

3 Medical Image Analysis Server

To overcome these problems a loosley coupled approach has been devised based on client-server communication architecture. A seperate medical image analysis server application will be specified and mechanisms for the relevant control and datapath are described.

The medical image analysis server approach is summarised in figure 3. Each of the boxes in this diagram represents an entirely seperate application being run on the host machine. The client is the commercial application, the browser is a standard web browser such as Netscape and the server is the OSMIA system specified in this document.

The interaction between these application is probably best understood by considering modes of operation of the system. Consider the following three types of user who may wish to use the system;

- Commercial developer. Developing software such as PnV and wishing to trial software in TINA in order to assess the suitability of the technique for inclusion.
- Algorithmic developer. Researching algorithms in TINA and wishing to interact with them using clinically appropriate applications.
- Clinical researcher. Experienced user of commercial application such as PnV wishing to try novel imaging techniques in TINA.

Each of these will make use of each part of the system but with varying degrees of capability in each area. For instance it is likely the clinical user will have little development experience and will rely on driving the system from the commercial application end. On the other hand the algorithmic developer will probably have little experience of the commercial application but will have good computing experience. In each case there will be subtle differences in their use of the system, however each user will follow the same basic pattern of use.

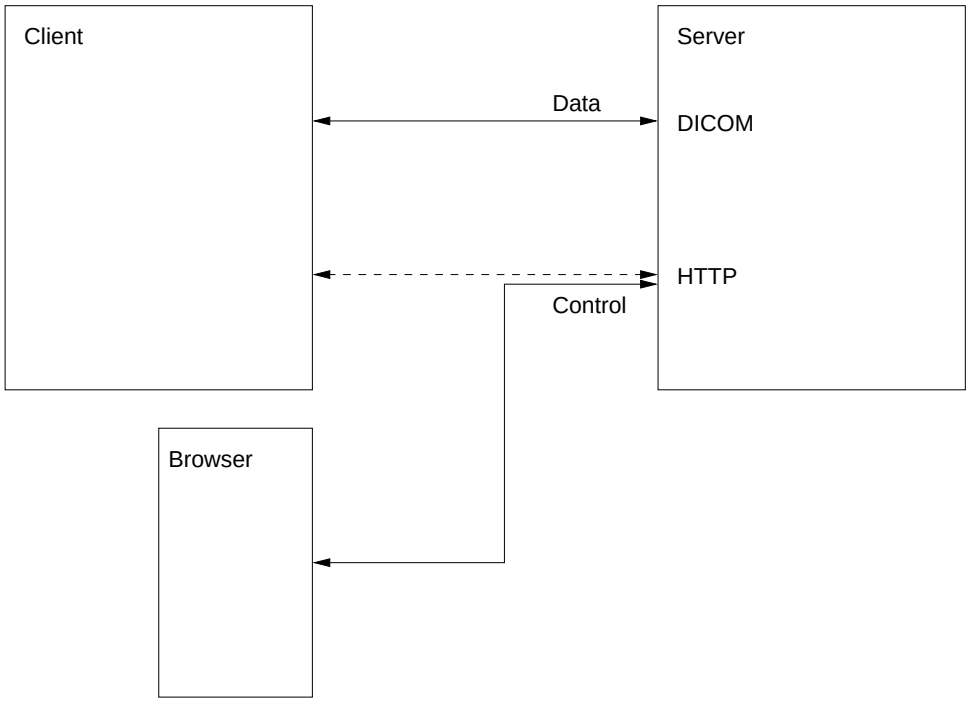

Figure 1: Medical image analysis server overview

The user would initiate the analysis server and connect to it using a web browser to query the provided functionality. From the commercial application the user would then transmit image data in DICOM format from the commercial application to the server. Depending upon the level of expertise the user may control the functionality of the server from the web frontend. In the case of a developer this may be achieved using a verbose and flexible connection to a web browser. Information returned via this channel may include intermediate results as well as debugging information. However, in the case of a clinical user this process may be replaced by a much simplier control mechanism potentially driven from the commercial application itself, which has been augmented with an inbuilt HTTP client (see diagram of figure 3). Once analysis is complete results are returned to the commercial application using the DICOM channel.

3.1 Benefits

A client-server approach represents a loosely coupled connection between TINA and the commercial application. This would provide much greater protection to the commercial application in terms of security and software robustness than directly linking TPS from TINA or another source (which may not conform the same level of coding standard as the main application). Data transfer is handled by an established medical standard, DICOM. The ability to send, recieve and handle DICOM data is a crucial component in any commercial medical image analysis system. Utilising this existing functionality in the client ensures the appeal of the server is not limited to applications which have been developed to work with it. Any application with DICOM push and recieve capability will be able to utilise the server functionality at some level. The use of HTTP as the protocol for control is motivated by similar intent. In the absense of any specific development of the commercial client application a generic webbrowser could be employed. If a more focused or automated user experience is required HTTP capabilities could be added to the commercial application, potentially making the use of the analysis server virtually invisible to the user.

The operating modes discussed so far have assumed that both the analysis server and the client(s) are to be executed on the same physical hardware. However, this need not be the case as both the DICOM and HTTP protocls which allow communication between the services are network protocols (that is are built upon OSI layers such as TCP/IP). Thus the server may be executing on remote hardware which may provide more compute power than the client machine or may be more appropriately located in terms of support (for instance at a site local to the algorithm developer).

4 Server Overview

The remainder of this document is concerned with the specification for the server side protocols.

A more detailed diagram of the server side of the interface is given below in figure 4. The server is a faceless process executing on the host machine awaiting connection from the client application. Externally the system is viewed as two servers which communication using either HTTP or DICOM.

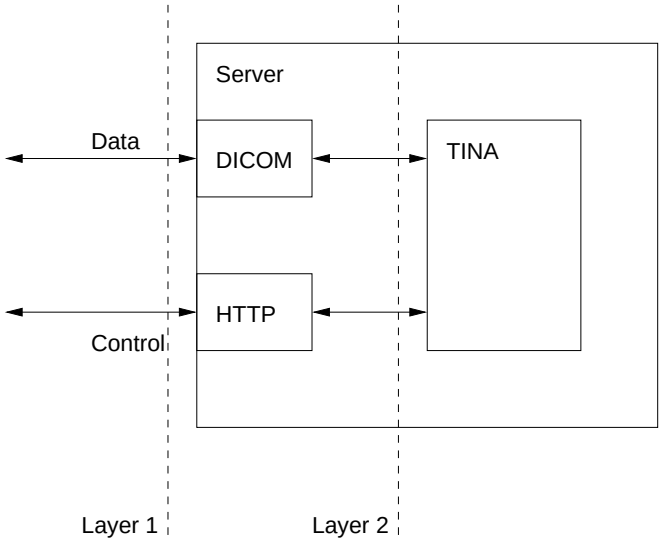

Figure 2: Server overview

The control layer is driven using HTTP. HTTP is the basic communications layer of the World Wide Web and is used to transmit HTML data from sites to web browsers. Although designed as an asymmetric transfer protocol (where bandwidth from server to client is greatest), requests from browsers can be constructed to include variables and thus duplex communication is achievable. The HTTP protocol can be found at [`http://www.w3.org/Protocols/HTTP/HTTP2.html`].

Image data is transmitted using the DICOM communications protocol. All major commercial medical image analysis and visualisation software is able to transmit and recieve DICOM. DICOM is almost always adopted as the protocol with which data is managed in hospital environments. DICOM is employed by all the major scanner manufacturers (CT, MR, PET and Ultrasound) as the method by which data is transfered electronically to PACS (Picture Archiving and Communication System) which form the basis of modern filmless radiological systems in hospitals. The DICOM standard definition can be found at [`http://medical.nema.org`].

The server has two important layers. The first is the external layer (layer 1 in figure 4). This layer uses the HTTP standard to control the server and the DICOM standard to request and supply data to and from the server. The second layer is the internal layer between the DICOM and HTTP subsystems and the processing itself (layer 2 in figure 4). This layer controls the transfer of data, commands and results to and from the TINA algoritms themselves. The architecture of this layer is largely dictated by the structure of TINA and utilises the already established control and data management stratergies. Despite this the final server framework will be sufficiently generic to allow its use with other TPS.

Issues

- Initial server will handle single users only, i.e. control and data are recieved from a single individual.
- Ultimately server could deal with several users either simultaneously or in a batch process method where processing ‘jobs’ are submitted.

5 Layer 1: External

This layer represents the interface between the OSMIA server and other applications. The aim of this part of the specification is to be 'broad', utilising the full potential of the employed protocols with as little limitation as possible. The specification must not be a esoteric prescription of the widely used HTTP and DICOM standards, it must encompass the 'spirit' of these standards. This way we can ensure that the server can be used by the largest number of HTTP or DICOM clients without modification.

5.1 Control

The control interface will use either the HTTP GET or POST mechanisms to construct variable encoded messages to the server. A GET request is used to retrieve data from the HTTP server and a POST request is used to send data to the HTTP server to be processed. However, data (such as HTML form data) can, and usually is, be sent to the server as URL-encoded, i.e. a request such as `http://osmiaserver.tina-vision.net/?name=tom&date=today` assumes the server at the address *osmiaserver.tina-vision.net* is expecting the fields *name* and *date* and assigns the values *tom* and *today* to these variables.

Issues

- The focus of this part of the document is on the specification of legal HTTP requests.
- It should define sets of variables which the server will accept and the contexts under which these are meaningful.
- Does this specification need to demand either GET or POST or can the system be built to use either? (GET is typically used to send small amounts of data).

Useful resources

- Good introduction to HTTP at [`http://www.jmarshall.com/easy/http/http_footnotes.html`].
- Open source C library for handling HTTP LibHTTPD [`http://www.hughes.com.au/products/libhttpd/`].

5.2 Data

Useful resources

- Open source DICOM toolkit DCMTK [`http://www.offis.uni-oldenburg.de/projekte/dicom/soft-docs/soft01_e.html`].

6 Layer 2: Internal

This layer represents the interface between internal HTTP and DICOM servers and the analysis backend, TINA. The system will utilise some of the fundamental data and control structures of TINA but will ensure that the constructed framework is generic enough for other uses. The details of this section have not yet been established.

6.1 Control

Issues

- Could utillise existing TINA tool control mechanisms? (ala TCL interface).

6.2 Data

Issues

- Decode of DICOM image data into Imrect
- Generation of new sequence for each dataset