ELSEVIER

NeuroImage

www.elsevier.com/locate/ynimg
NeuroImage 24 (2005) 1170–1179

The LONI Debabeler: a mediator for neuroimaging software

Scott C. Neu,[a,*] Daniel J. Valentino,[a,b] and Arthur W. Toga[a]

[a]*Department of Neurology, UCLA Laboratory of Neuro Imaging, David Geffen School of Medicine, Room 4-238, 710 Westwood Plaza, Box 951769, Los Angeles, CA 90095-1769, USA*
[b]*Department of Radiological Sciences, UCLA, Los Angeles, CA 90095-1721, USA*

Received 11 June 2004; revised 12 September 2004; accepted 28 October 2004
Available online 31 December 2004

Brain image analysis often involves processing neuroimaging data with different software packages. Using different software packages together requires exchanging files between them; the output files of one package are used as input files to the next package in the processing sequence. File exchanges become problematic when different packages use different file formats or different conventions within the same file format. Although comprehensive medical image file formats have been developed, no one format exists that satisfies the needs of analyses that involve multiple processing algorithms. The LONI Debabeler acts as a mediator between neuroimaging software packages by automatically using an appropriate file translation to convert files between each pair of linked packages. These translations are built and edited using the Debabeler graphical interface and compensate for package-dependent variations that result in intrapackage incompatibilities. The Debabeler gives neuroimaging processing environments a configurable automaton for file translation and provides users a flexible application for developing robust solutions to translation problems.

Keywords: File format; Translation; Automated analysis; Software integration

Introduction

Communication between neuroimaging software that was written by different authors is often achieved through the exchange of image data files. These files contain volumetric or surface data along with additional descriptive information (metadata). For example, consider image data acquired in the coronal plane by a magnetic resonance scanner and stored using the DICOM[1] (Bidgood and Horii, 1992; Bidgood et al., 1997) file format. In order to perform a particular analysis, it may be necessary to convert the data to the ANALYZE[2] (Robb et al., 1989) file format to use it as input for one software package, and then convert the output to the MINC[3] file format for a second package. An entirely new analysis may require reformatting the image data into the axial plane and converting the data into other file formats specified by other image processing and analysis packages (Rex et al., 2003; Fissell et al., 2003).

Most neuroimaging software packages can be classified as monolithic or modular environments (Rex et al., 2003). Monolithic environments offer a complete and unified analysis in one package whereas modular environments enable users to configure their own sequences of analysis modules. Monolithic environments usually ensure that each processing step is compatible with the next by strictly limiting the structure of the information passed between steps. Although this improves runtime stability, it inhibits the integration of the system with software packages that do not adhere to the same structural restrictions. It also makes it difficult to perform analyses on neuroimaging data stored in incompatible file formats and to exchange data between institutions. Because it seems unlikely that the efforts of the brain mapping community will standardize on a single file format and set of conventions due to the different and changing needs of individual researchers, it ultimately becomes necessary to develop solutions to the interoperability problems encountered in modular environments.

Neuroimaging file formats currently use one or more files to store image data and its associated metadata. The two most common configurations are illustrated in Fig. 1. In Fig. 1a, the metadata is stored at the beginning of one file and is followed by the image data (in some cases, additional metadata may follow the image data as well). This paradigm is used by the DICOM and MINC file formats, and is allowed by the Interfile (Todd-Pokropek et al., 1992) and NIFTI[4] file formats. Most often, the DICOM file format will store each slice of an image volume separately, which results in many files of this type. The second configuration is shown in Fig. 1b, which stores the metadata in one file and the image data in a different file. The ANALYZE and AFNI[5] (Cox, 1996) file formats use this two-file paradigm, which is again

* Corresponding author. Fax: +1 310 206 5518.
E-mail address: neu@loni.ucla.edu (S.C. Neu).
Available online on ScienceDirect (www.sciencedirect.com).
[1] http://medical.nema.org/dicom/2003.html.
[2] http://www.mayo.edu/bir/PDF/ANALYZE75.pdf.
[3] http://www.bic.mni.mcgill.ca/software/minc/minc.html.
[4] http://nifti.nimh.nih.gov/dfwg.
[5] http://afni.nimh.nih.gov/pub/dist/doc/README.attributes.

1053-8119/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.neuroimage.2004.10.035

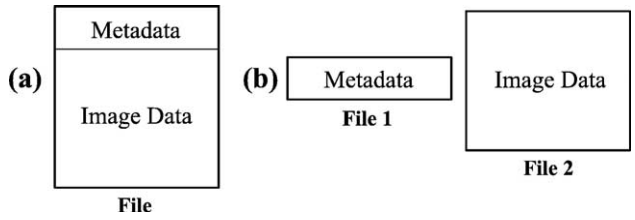

Fig. 1. Two common paradigms used by neuroimaging file formats. (a) One file contains both the metadata and image data, with the metadata stored at the beginning of the file. (b) One file contains the metadata and a second file contains the image data. We refer to the file in (a) and the two files in (b) as "ordered file sets".

allowed by the Interfile and NIFTI file formats. Because the number of files varies between different file formats and these files are logically considered to be a whole, we refer to the files used in either of the two paradigms as an "ordered file set" (even if only one file is used, as in Fig. 1a).

Many neuroimaging file formats (e.g., ANALYZE, Interfile, AFNI, and MINC) were defined by researchers to meet the specific needs of scientific communities, and this tends to limit the ability of end-users to access, analyze, and share data. Although the DICOM file format was developed for the open, standard-based exchange of biomedical imaging data (and is currently used on nearly all medical imaging devices), device vendors typically use different terminology and conventions in their implementations. The problems encountered with multiple file formats while attempting to integrate software modules that acquire, process, and display data in a biomedical data processing environment have been recognized for many years (De Cuyper et al., 1991). In fact, the problem of automatically producing a mapping between elements of two schemas has been largely studied in areas such as electronic commerce, data warehousing, and web-oriented databases (Rahm and Bernstein, 2001). However, most efforts to address software integration in the neuroimaging community have concentrated on the introduction of additional file formats (further complicating the problem) and the development of languages[6] that describe file formats (De Cuyper et al., 1991). Although such languages provide information useful for converting between file formats, they do not address the larger issue of how to perform the conversions themselves. Wiederhold (1992) has suggested that it is necessary to introduce a software layer between data sources and applications in order to simplify, abstract, reduce, merge, and explain the data. This effectively decouples data from the applications through the introduction of mediators that make decisions by choosing and evaluating pieces of information.

Metadata may be encoded in an ordered file set in many different ways. To illustrate this, we have taken the metadata shown in Table 1 and encoded it using two hypothetical file formats in Fig. 2. The metadata consists of three "metadata elements", each having a name ("patient id", "series date", and "echo time") and a value ("12345", "January 8, 1953", and "82 milliseconds"). The hypothetical file format in Fig. 2a associates a numerical term with each element name (e.g., "00100020" is associated with "patient id") and defines how to encode each element value (e.g., "January 8, 1953" is encoded as "01/08/1953"). It is worth noting that some neuroimaging file formats (e.g., DICOM) do not require some metadata elements to be present in their files, and they also allow

[6] http://forge.gridforum.org/projects/dfdl-wg.

Table 1
Three metadata elements

Name	Value
Patient id	12345
Series date	January 8, 1953
Echo time	82 ms

The values of these elements are encoded in two different ways in Fig. 2.

the existence of elements whose names are defined by individuals for private use. The second hypothetical file format in Fig. 2b relies upon the order of the encoded values to determine their meaning; the first value ("12345") always refer to the patient id and the second value ("19530801") always refer to the series date. As is true with many fixed-size neuroimaging file formats, this hypothetical file format does not define places to store all the metadata (in this case, the echo time). In practice, this leads to information loss when multiple file format conversions are used. Although some researchers prevent information loss by storing needed information in unused places (e.g., replacing "19530801" with "82" if the series date can be ignored but the echo time cannot), this often leads to confusion when end-users obtain their files without knowing about the replacements.

Both file producers (creators of neuroimaging files) and file consumers (readers of these files) contribute to the problems experienced by neuroimaging software users. The incompatibility of two software packages is often due to variations in the metadata encoded in the files exchanged between them. For example, many of these variations are introduced by the file producers, who:

- Decide where in the file to encode information.
- Choose how much information to encode in the file.
- Define the nomenclature and terminology used.
- Can add information and use encodings not defined by the file format.
- May incorrectly encode file information.

This leads to problems when file consumers:

- Decide what types of neuroimaging data to support.
- Rely on the presence of information and use of expected terminology.
- Need to order and correlate information in predetermined ways.

Many of these variations occur because the encoding rules defined by a file format often do not guarantee the existence of certain metadata elements nor force file producers to use suggested

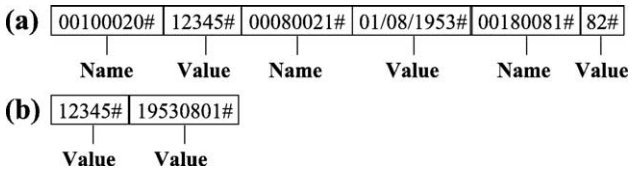

Fig. 2. Two different ways to encode the metadata in Table 1. (a) A numerical term is associated with each element name (e.g., "00080021" is associated with "series date") and its value is encoded afterwards (e.g., "January 8, 1953" is encoded as "01/08/1953"). (b) The values of two elements (patient id and series date) are encoded and their sequential order determines their meaning. No encoding is defined for the third element (echo time).

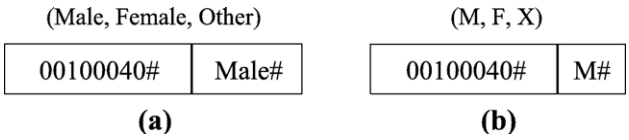

Fig. 3. Different terminology used with the same encoding rules. (a) The terms "Male", "Female", and "Other" denote the patient's gender. (b) The symbols "M", "F", and "X" denote the patient's gender. The name/value encoding from Fig. 2a is used in both (a) and (b), but differences in the encoded values are allowed.

nomenclature and terminology. Realistically, only metadata elements used while file decoding can be made to follow the definitions of a file format. For example, if the encoded image width and number of bits per pixel were incorrect, the decoding process would most likely fail. Consequently, file producers are forced to adhere to the encoding rules for these elements and may not use any values different from those defined by the file format. Metadata elements that are not used during the decoding process tend to give descriptive information about the image data (e.g., minimum or maximum pixel values), the patient (e.g., patient id or gender), and acquisition (e.g., echo time or series date). Since these elements are not used while file decoding, file producers have the freedom to choose their values. They may use conventions suitable for their own purposes and define terminology meaningful only to their current context.

In order to manage these variations, it is important to conceptually separate the encoding rules defined by a file format from the terminology and conventions used to describe metadata. Fig. 3 illustrates two cases where the gender of a patient (represented by "00100040") is encoded as a name/value pair (see Fig. 2a). In Fig. 3a, the terms "Male", "Female", and "Other" are used to denote the patient's gender whereas in Fig. 3b the symbols "M", "F", and "X" are used. In both of these cases the same encoding rules are used (name followed by value), but the terminology is different. As previously stated, it is not possible to force file producers to use a particular convention. File producers usually deviate from suggested terminology and conventions for two reasons. First, neuroimaging information is by nature complicated and subjective to what is being done and for what purpose[7] (Wiederhold, 2003). This makes it difficult to standardize element names and values when exceptions are commonly found. Second, authors tend to be too restrictive with some of their definitions and too general in others. When a definition is too restrictive, file producers may be compelled to redefine it or even replace it with an entirely different definition. When a definition is too general or made optional, file producers may add information however they choose, and this results in file consumers seeing unwanted variations (e.g., "Male" and "M") in the metadata.

Because many of the metadata variations found in neuroimaging files depend upon the terminology and nomenclature used during the creation process, knowledge of the file formats alone is not enough to perform file format conversions. Since file producers (e.g., scanner manufacturers or software packages) choose how to add variations, and file consumers (e.g., software packages or image viewers) decide which variations to support, it is necessary to build file format conversions between different file producers and file consumers.

[7] http://www.informatik.uni-trier.de/~ley/db/journals/omics/omics7.html.

Methods

We have built the LONI Debabeler to mediate file exchanges between neuroimaging software packages using a set of producer/consumer-dependent translations. The Debabeler is a flexible tool for creating, editing, and managing translations between neuroimaging file formats. It uses a framework that automates the selection of a translation for each set of input files. This enables users to use the Debabeler to translate different file formats without intimate knowledge of metadata variations. It also provides an automated method of translating files between two or more neuroimaging software packages within data processing environments (Rex et al., 2003; Fissell et al., 2003). The Debabeler graphical interface enables users to manually select and execute translations as well as generate independent executables that translate on their own.

The process used by the Debabeler to translate input files is illustrated in Fig. 4. The first step involves choosing the input files and selecting a "target". The target specifies the output file format and the file consumer that will receive the translated files. For example, the target "ANALYZE Files for AIR" could define Debabeler output written using the ANALYZE file format for input to the software package AIR. The second step shown in Fig. 4b arranges the files into ordered file sets (see Fig. 1). The two

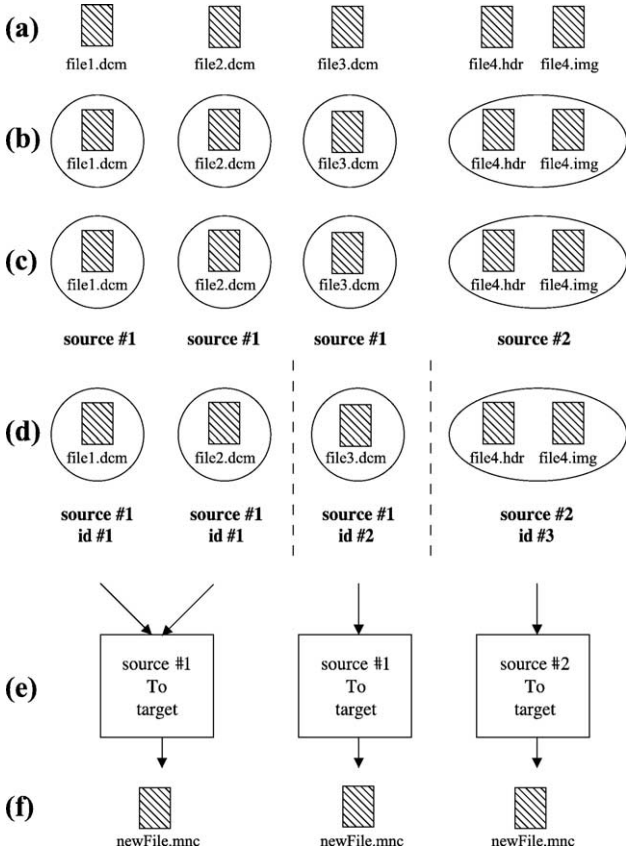

Fig. 4. Steps used by the Debabeler to translate files. (a) The input files are selected and an output target is chosen. (b) Files that are collectively defined by a file format are arranged into ordered file sets. (c) The producer of each ordered file set is identified. (d) A group id is assigned to each ordered file set. (e) A translation is found for each group that converts the files in the group into the target file format. (f) Each group of ordered file sets is individually translated.

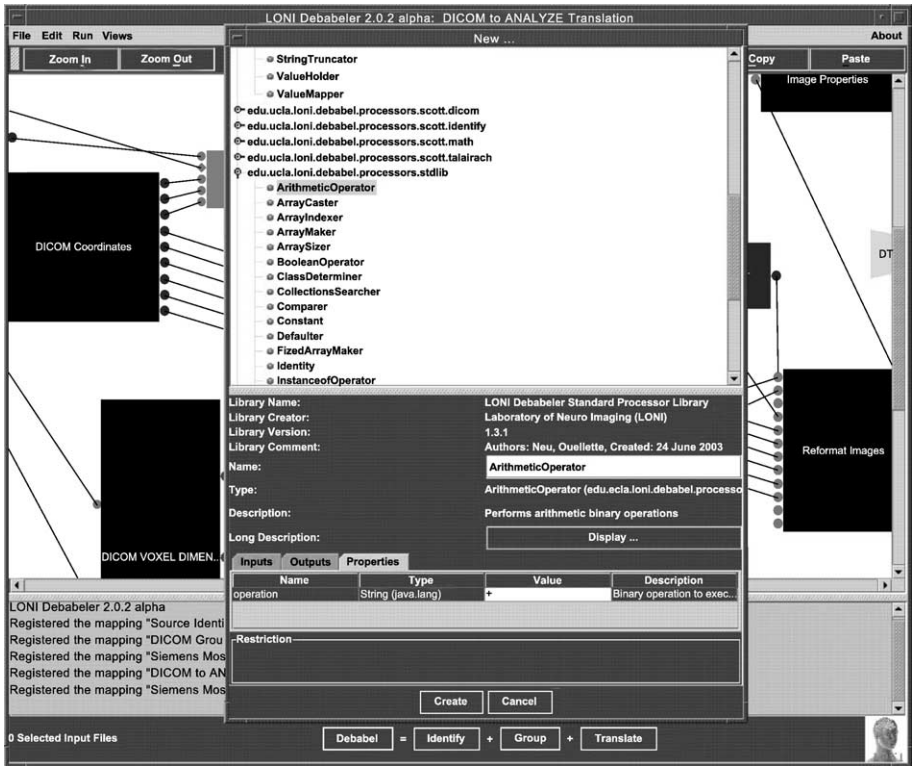

Fig. 5. The Debabeler GUI while selecting a new processor. The dialog in the foreground lists the available processors as well as editable information about each processor's function, inputs, and outputs. New processors are created in the window behind the dialog, where they can be graphically connected to other functional modules.

ANALYZE files belong to one set and each DICOM file has its own set. The next step identifies the file producer that created each ordered file set. In Fig. 4c, the first three files are identified as having been produced by "source #1" and the last file set is identified as having been produced by "source #2". Common file producers include medical image scanners (e.g., fMRI images acquired at a Siemens scanner and stored in a mosaic layout using the DICOM file format) or the output files produced by software packages. The step shown in Fig. 4d further labels each ordered file set by assigning each set a group id. These group ids are used to form groups of ordered file sets that logically belong together. This can be used to separate proton density MR image files from their T2 counterparts or to remove scout images from an image series. Once each ordered file set has been identified and grouped, each group is translated independently from the others. The Debabeler uses the group and target names to find the appropriate translation for each group from its translation registry, and then uses the translation to convert the files into the target file format. The three groups shown in Fig. 4d are translated into the files shown in Fig. 4f using the two translations in Fig. 4e.

Visual environment

Solutions to translation problems can be developed within the Debabeler's visual environment. Its Java graphical user interface (Fig. 5) provides access to libraries of functional modules ("processors") that can be graphically connected together between input and output trees of image data and metadata. This configurability enables users to provide missing information (e.g., minimum and maximum pixel values) and correct erroneous metadata values, as well as expand abbreviations (e.g., change "M" to "Male"), decipher enumerations (e.g., change "5" to "SPGR"), and convert between units (e.g., Gauss to Tesla). With the appropriate processors, proprietary metadata may be decoded and conversions can be applied between different pixel data types (e.g., convert 1-byte pixel values to 2-byte pixel values).

Fig. 6a shows an example of two connected modules. The module on the left has one input and two outputs, with one output connected to an input of the module on the right. A "macro-module" encapsulates modules and the connections that join them. The two processors and their connection in Fig. 6b are contained in the macro-module of Fig. 6a. Every unconnected input or output inside a macro-module is an input or output of the macro-module itself unless specified otherwise; for example, the inputs of the macro-module in Fig. 6a are the inputs of the first processor in Fig. 6b.

The LONI Debabeler uses the Java 1.4 Image I/O architecture[8] to read and write neuroimaging files. Because image data and metadata are organized and managed as trees, Debabeler users are isolated from the underlying encoding rules of different file formats. Another benefit of the Java 1.4 Image I/O architecture is its capability to register plugins that decode and encode files. Authors who write file decoders and encoders for new neuroimaging file formats can add new functionality to the Debabeler by adding their plugins to the Debabeler runtime classpath. We have extended the plugin functionality to Debabeler processors so that it is possible for authors to add processor libraries for new manipulations of metadata and image data.

[8] http://java.sun.com/j2se/1.4.2/docs/guide/imageio/index.html.

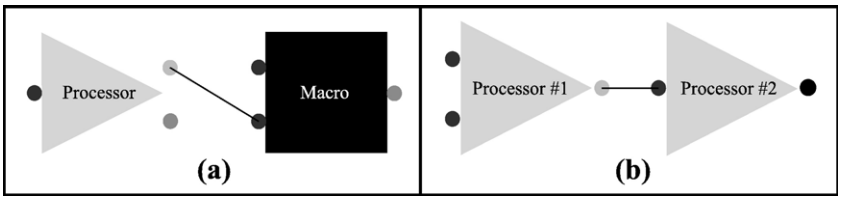

Fig. 6. (a) Two connected modules. The triangular-shaped module is a processor that has one input and two outputs, and the square-shaped module is a macro-module that has two inputs and one output. A connection joins an output of the processor with an input of the macro-module. (b) The modules and connection contained within the macro-module of (a). The unconnected inputs and outputs of the modules are the input and output nodes of the macro-module.

Translations are read from and written to disk as XML files and can be exchanged between two users as long as the processor libraries of both users are the same. A very convenient feature of the Debabeler is its capability of combining translation XML files and processor libraries into an executable Java jar file. Each executable jar file runs independently and contains everything needed to perform translations. They are started from the command line and require specification of the input files, output target, and optional translation arguments. Because they are written in Java, the executable jar files can be installed on web servers to deliver translation functionality to remote clients via their web browsers.

Debabeling

The steps of identifying the producer of an ordered file set (Figs. 4b–c), assigning a group id to the set (Fig. 4d), and translating the set into its output files (Figs. 4e–f) are collectively termed "debabeling". These identification, group assignment, and translation tasks are accomplished by making connections from input trees of metadata and image data to output trees using processors from the Debabeler libraries.

Identification

The Debabeler first determines the file format associated with each input file. This is accomplished either by inspecting the format's "magic number" (often a character string embedded at the beginning of a file; e.g., "DICM" for the DICOM file format) or by recognizing the format's file name extension (e.g., ".dcm"). After files that collectively define a file format are arranged into ordered file sets, the Debabeler identifies the file producers who created them. One such identification is shown in Fig. 7a. The input tree contains the file set names, the name of the Debabeler decoder that was able to decode the file set, and the file metadata. The target tree receives the name of the file producer (in this case, "siemens mosaic dicom"). The logic used to determine the file producer name is shown in Fig. 7b. Here the metadata is parsed to find an image type element, and the element value is examined for the keyword "MOSAIC". Likewise, the scanner manufacturer value is examined for the keyword "SIEMENS". If these two searches yield positive results, the Debabeler identifies the ordered file set as a DICOM file created by a Siemens scanner whose images are arranged in a mosaic layout. Otherwise, if the Debabeler DICOM decoder was used, the file set is identified as a standard DICOM file ("standard dicom"). The identification step is a process that identifies file producers by decoding file formats and searching for specific metadata values. As such, it is capable of automatically separating a mixture of different file types.

Group assignment

Once an ordered file set is identified, the Debabeler assigns it a group id. Group ids determine what file sets are grouped together

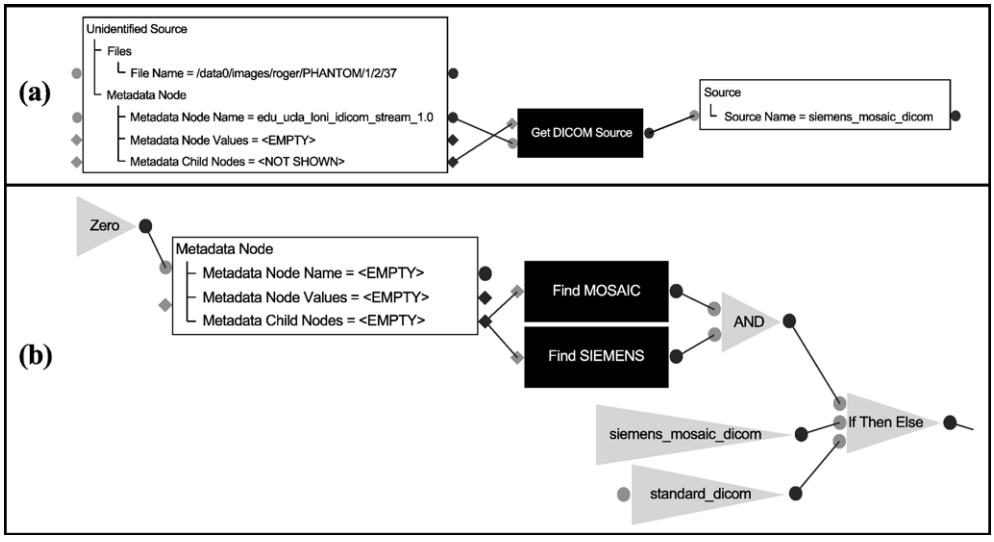

Fig. 7. Identifying two different DICOM file producers. (a) An input tree contains the file name, format, and metadata of the file being identified. The macro-module "Get DICOM source" uses this information to determine the name of the file producer ("siemens mosaic dicom"), which is sent to the output tree. (b) Logic contained inside the macro-module of (a) involves searching for the keywords "MOSAIC" and "SIEMENS" in order to distinguish between a "standard" DICOM file and one produced by a Siemens scanner with a mosaic image layout.

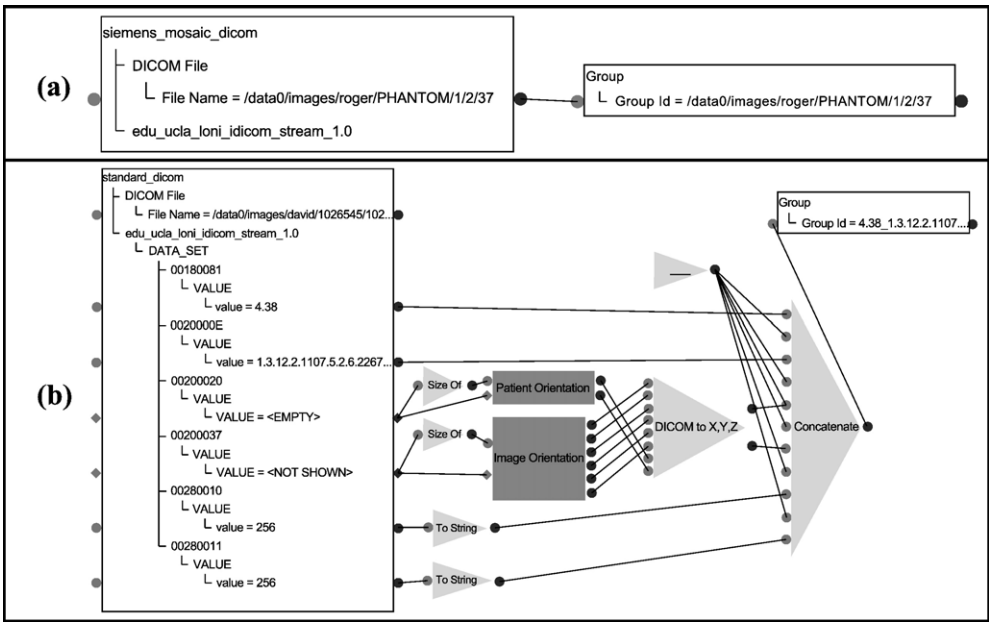

Fig. 8. Two types of DICOM group assignments. (a) The group assignment for a Siemens mosaic DICOM file uses the file name as the group id since all the images of the series are contained inside the file. (b) The group assignment for standard MR DICOM files uses the echo time, unique series id, patient orientation, image orientation, image width, and image height to generate a group id.

during translation. The group assignment step allows users to define how neuroimaging files are organized into image series. This is advantageous when a file producer imposes an ordering that needs to be altered (e.g., a scanner that combines scout images with image slices) or when computational limitations require more manageable groups of files. A group assignment that assigns group ids to DICOM files with a mosaic image layout produced by a Siemens scanner is shown in Fig. 8a. Since all of the images in the series are contained in one file, the name of the DICOM file is used as the group id. This ensures that all the images in the DICOM file are translated as one group. For MR DICOM series that store one image slice per file, a different group id is needed to group all the images in the series together. Fig. 8b shows the processors involved in generating a group id that consists of the echo time [(00018,0081)], series id [(0020,000E)], patient [(0020,0020)] and image [(0020,0037)] orientation, and the image width [(0028,0010)] and height [(0028,0011)]. Inclusion of echo time separates proton density from T2-weighted images, and slice orientation separates images that were acquired in different orthogonal planes. Assigning the same group id to DICOM files from the same image series groups the files together during the translation step.

Translation

After the identification and group assignment steps, the Debabeler translates each group of ordered file sets. Using the name of the file producer and the output target, an appropriate translation is found for each group and used for file conversion. Fig. 9 depicts two translations that convert Siemens mosaic DICOM files and standard MR DICOM files into ANALYZE files. In each translation, the image slices are reformatted to comply with the SPM[9] extension rules. In the Siemens mosaic DICOM to ANALYZE translation in Fig. 9a, a special processor is used to decode Siemens' private DICOM element (0029,1020) to obtain information about the mosaic layout. This information is used to divide the mosaic image into its components, and the component images are sorted into an image array. If the image volume is not consistent with the SPM conventions, it is reformatted into the correct orientation. Similar operations are performed in the standard MR DICOM to ANALYZE translation in Fig. 9b. However, since there are many DICOM files in a single group, the metadata and image data from all the files must be collected and combined. Fig. 9c shows the output tree for both translations where the metadata and image data for the output ANALYZE files are set.

[9] http://www.fil.ion.ucl.ac.uk/spm/distrib.html.

Results

Table 2 lists the encoder and decoder plugins currently supplied with the Debabeler. Although the Debabeler may be used to create file format translations from scratch, many users may prefer to edit the translations (Table 3) that are supplied with the Debabeler download package. These translations can be used without modification if they are appropriate, but in most cases users will want to edit them (e.g., change the output file name) to better satisfy their needs. We also include the Java source code for our processor libraries so that expert users may better understand and fix technical problems.

The results obtained by running a Debabeler executable Java jar file on a LINUX computer (1 Gb of RAM, one CPU at 1.8 GHz) to convert different numbers of DICOM files to the NIFTI file format are summarized in Table 4. Each of the five rows of the table corresponds to converting unsorted DICOM files into one or more NIFTI files and recording the total amount of time taken to perform the conversion. In each case, the Java virtual machine

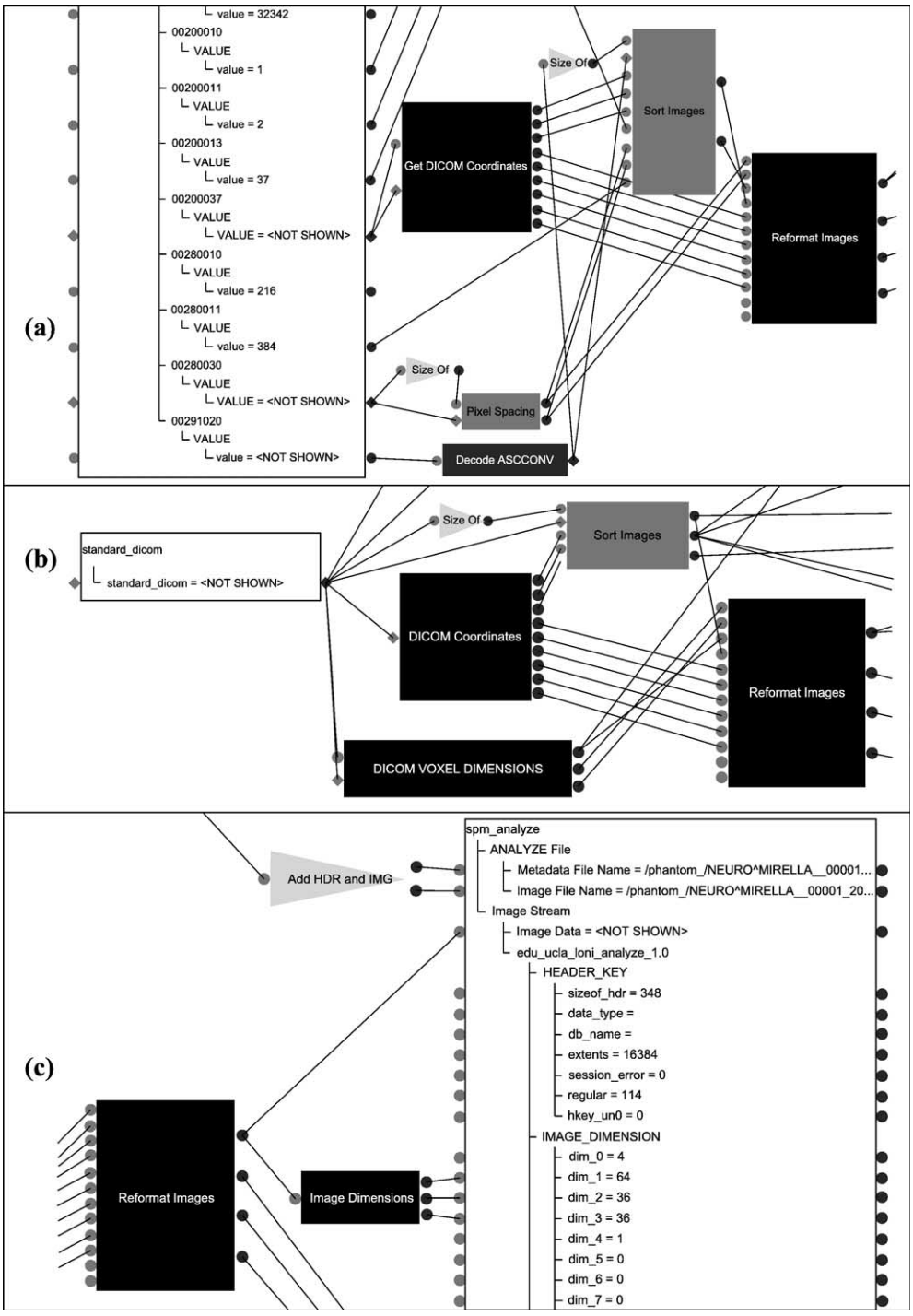

Fig. 9. Two DICOM translations. (a) Metadata from a Siemens mosaic DICOM file is used to divide the mosaic image into its component images. These images are sorted and reformatted into an orientation consistent with SPM conventions. (b) Images from multiple MR DICOM files are similarly sorted and reformatted. (c) Metadata and image data are stored in the ANALYZE file format. This occurs in both translations.

that ran the executable jar file was allocated 200 Mb of memory. As can be seen from Table 4, the DICOM file conversion rate remained relatively constant (7–8 files/s) as the number of DICOM files increased since the Debabeler first assigns each DICOM file to a group, and then translates each group separately from the others.

In addition to mediating file exchanges, the Debabeler is a useful tool for performing common neuroimaging file tasks. One such task is the removal of patient-identifying information (anonymization) from an image file. An example of an anonymization that de-identifies a standard MR DICOM file is shown in Fig. 10a. To ensure that no patient information is present in the name of the anonymized DICOM file, the original DICOM file name is completely replaced with a file name that uses current date and time. All nonstandard DICOM elements (proprietary metadata defined by scanner manufacturers) are removed from the original

Table 2
Neuroimaging file formats supported by the Debabeler Java 1.4 Image I/O plugins

File format	Read (decode)	Write (encode)
AFNI	✔	✔
ANALYZE	✔	✔
DICOM	✔	✔
GE	✔	✔
MedX	✔	
MINC	✔	✔
NIFTI	✔	✔
Raw	✔	✔

file and all unique ids (possibly containing unique scanner codes) are one-way encrypted to retain uniqueness. Further, a command line argument is used to replace the patient id in the anonymized DICOM file. The Debabeler's programmable interface allows users to construct different levels of patient de-identification in order to satisfy different standards of patient confidentiality (e.g., HIPAA). Fig. 10b shows an example of how the Debabeler can be used to convert a raw (no encoded metadata) neuroimaging file into a standard neuroimaging file (ANALYZE). In this case, the Debabeler reads the file as though it were a single image block, and command line arguments (e.g., image width, height, and color type) are used to divide the image block into multiple images. This can also be used to skip past fixed-sized metadata blocks if the sizes of the blocks are known. As shown in Fig. 10c, the Debabeler is capable of writing to more than one file format in a single translation. Here the images decoded from a MINC file are reformatted and stored as an ANALYZE file. Since the ANALYZE file format limits encoded metadata to 348 byte and does not define temporal scanner metadata, the repetition, echo, and inversion times in the MINC file are instead stored in an XML file. Using multiple file formats in translations can prevent important metadata from getting lost, and using XML saves them as human-readable text.

Discussion

Neuroimaging data are acquired by imaging devices and processed by software packages, and both store image intensity values and associated descriptive information to files using a variety of file formats. Although some file producers create files using the same file format, many use different terminology and nomenclature for the descriptive data. This becomes problematic for biomedical image analysis programs that expect either a specific file format or depend upon particular metadata conventions. Despite the development of "standard" medical image file formats, no one format has satisfied the needs of image-based studies involving multiple processing and analysis algorithms.

Table 3
Translations (T) and anonymizations (A) supplied with the Debabeler

Source	Target file format				
	AFNI	ANALYZE	DICOM	MINC	NIFTI
ANALYZE	T	A	T	T	T
DICOM	M	T	A	T	T
GE	M	T	T	T	T
MINC	M	T	T	A	T
Raw	M	T	M	M	M

An anonymization removes patient-identifying information from a file. Multiple translations (M) can be used to convert between file formats that do not have a single translation.

Table 4
Time taken by the Debabeler to convert DICOM files to NIFTI files

DICOM files (input)	Conversion time (s)	Conversion rate (file/s)	NIFTI files (output)
123	20	6.2	1
219	31	7.1	3
354	51	6.9	9
668	87	7.7	14
1207	152	8.0	28

Five sets of DICOM files were converted to NIFTI files on a LINUX computer (1 Gb of RAM, one CPU at 1.8 GHz).

The LONI Debabeler is a flexible tool that manages and converts a variety of biomedical image file formats. Its general architecture is not limited to specific file formats or metadata conventions, and it can be readily extended to decode (read) and encode (write) new file formats. The Debabeler converts input files by identifying the producer of the file, arranging the files into groups, and translating each group into a specified target format. This conversion methodology enables the Debabeler to operate as a mediator between neuroimaging software packages that use different file formats and rely upon different metadata conventions. The Debabeler's graphical user interface enables users to add and modify translations to meet their specific needs. The Debabeler thus serves as an intelligent translation manager that automatically selects translations and converts common neuroimaging file formats in flexible ways, and a graphical development environment for visually programming solutions to translation problems.

Although the Debabeler represents the input and output file formats as trees (hiding the details of how the data are encoded in the files), in many cases expert knowledge is needed to properly map input values to output values. For example, if the values for the image width and height are not correctly assigned in the output tree, the Debabeler will create a file that most likely will be undecodable. This is because the values of the image width and height are usually used to decode the image data in the file, and if they are incorrect the image data will not be properly decoded. To address this issue, we include translations along with the Debabeler that perform conversions between many common file formats (see Table 3). It has been our experience that most translation requests are for editing existing translations (reformatting a string field, changing the output file name, etc.), and we have found that the Debabeler's graphical interface provides a quick and easy way to adjust these translations.

Another concern is that in an environment of N software modules, one may need to program $N \times (N - 1)$ translations to ensure interoperability between each pair of modules. To this end, we remark that it is not always the case that each module will have vastly different file format requirements and we can expect some modules to work together without Debabeler mediation. In fact, the more translations that exist, the more likely a translation can be found for a newly introduced module. Furthermore, we note that the identification step used by the Debabeler is configurable and generalizable. If two different file producers create input files that share the same file format and use similar metadata conventions

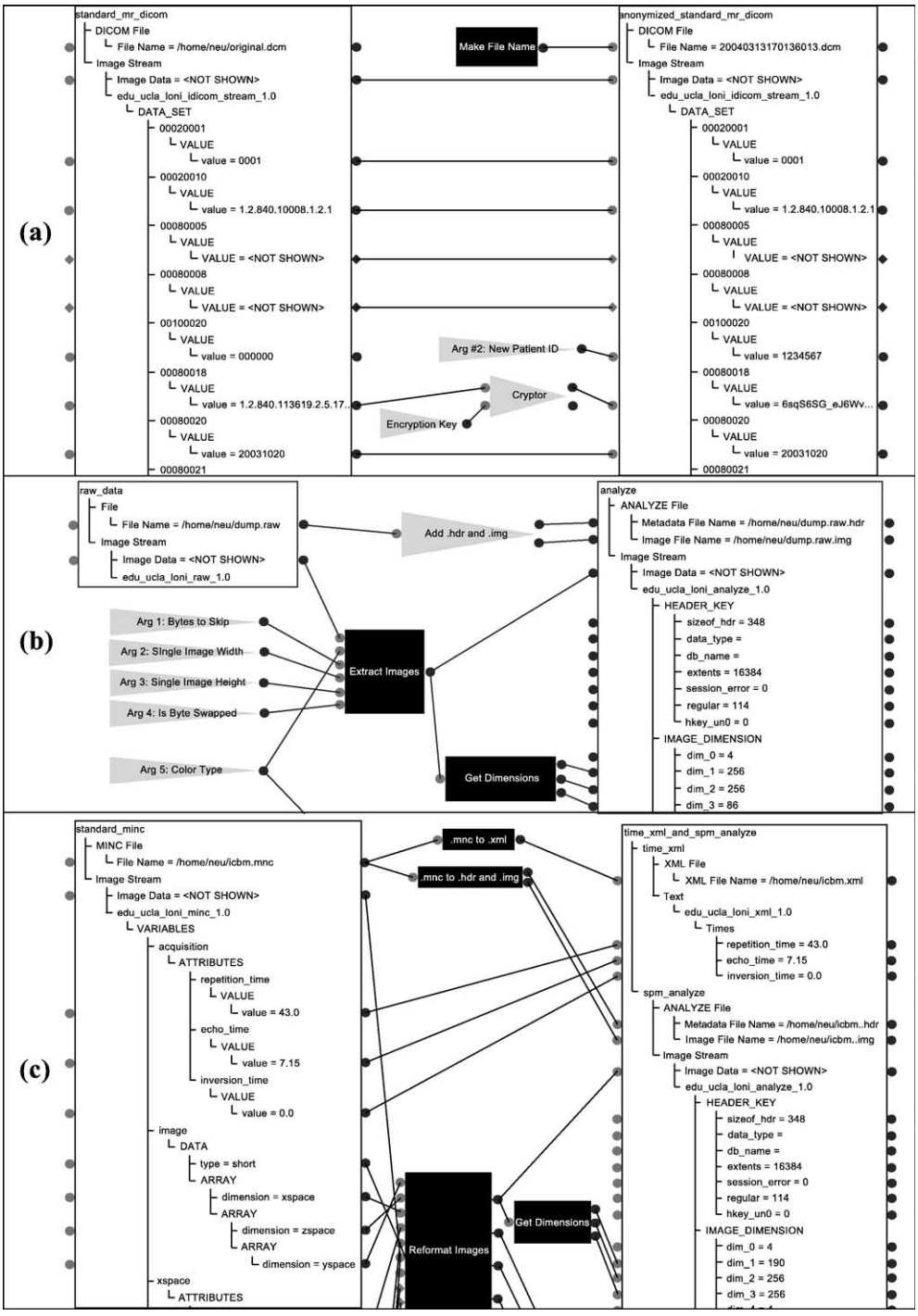

Fig. 10. Additional Debabeler uses. (a) A DICOM file is anonymized by removing patient-identifying information, replacing the patient id, and encrypting unique ids. (b) A block of raw input data is divided into multiple images and written to an ANALYZE file. (c) The images decoded from a MINC file are written to an ANALYZE file and extratemporal metadata is saved to an accompanying XML file.

and terminology, it is possible (and preferable) to identify the files as being produced from a common file producer and account for their differences in the same translation. This greatly reduces the number of translations needed and also adds to the robustness of each translation.

The LONI Debabeler is available from http://www.loni.ucla.edu/Software using the search key word "Debabeler".

Acknowledgments

This work was generously supported by a research grant from the NIMH and NINDS (P20 MH065166), and a resource grant from the NCRR (P41 RR013642), as well as the Biomedical Informatics Research Network. Additional support for data collection was from the NIBIB, NINDS, and NIMH (P01 EB001955).

References

Bidgood, W.D., Horii, S.C., 1992. Introduction to the ACR-NEMA DICOM standard. Radiographics 12 (2), 345–355.

Bidgood, W.D., Horii, S.C., et al., 1997. Understanding and using DICOM, the data interchange standard for biomedical imaging. J. Am. Med. Inform. Assoc. 4 (3), 199–212.

Cox, R.W., 1996. AFNI: software for analysis and visualization of functional magnetic resonance neuroimages. Comput. Biomed. Res. 29 (3), 162–173.

De Cuyper, B., Nyssen, E., et al., 1991. Do you also have problems with the file format syndrome? Med. Biol. Eng. Comput. 29 (6), 55–60.

Fissell, K., Tseytlin, E., et al., 2003. Fiswidgets: a graphical computing environment for neuroimaging analysis. Neuroinformatics 1 (1), 111–126.

Rahm, E., Bernstein, P.A., 2001. A survey of approaches to automatic schema matching. VLDB J. 10 (4), 334–350.

Rex, D.E., Ma, J.Q., et al., 2003. The LONI pipeline processing environment. NeuroImage 19 (3), 1033–1048.

Robb, R.A., Hanson, D.P., et al., 1989. Analyze: a comprehensive, operator-interactive software package for multidimensional medical image display and analysis. Comput. Med. Imaging Graph. 13 (6), 433–454.

Todd-Pokropek, A., Cradduck, T.D., et al., 1992. A file format for exchange of nuclear medicine image data: a specification of Interfile version 3.3. Nucl. Med. Commun. 13 (9), 673–699.

Wiederhold, G., 1992. Mediators in the architecture of future information systems. Computer 25 (3), 38–49.

Wiederhold, G., 2003. The impossibility of global consistency. Omics: J. Integr. Biol. 7 (1), 17–20.