POSDA Overview

Bill Bennett, May 5, 2008

Introduction

This document is an overview of POSDA. At the end of this document, there is a list of other documents which provide information on how to install and use POSDA.

POSDA stands for "Perl Open Source Dicom Archive", but I'm thinking of changing the "A" to "Analytics", because it is primarily good for analyzing DICOM files, or, analyzing collections of DICOM files. [I've also thought of changing the "A" to the more generic "Acronym", but ...,]

POSDA is set of `perl` modules for reading, writing, and manipulating DICOM data sets. It's a handy toolkit for manipulating DICOM format files in `perl`. Its good for analyzing the contents of groups of DICOM files, for making batched changes to DICOM format files, etc. POSDA is written in "pure `perl`" (i.e. it doesn't rely on any packages written in other languages like C or C++), and relies only on standard modules (except for `DBD:Pg` to interface to `postgresql`), so it should be pretty easily portable to most UNIX or UNIX-like environments (Windows may be sufficiently UNIX-like. Lots of people assure me that it is, but I'm not convinced. But, hey, I use a Mac).

POSDA was developed under Linux, and MacOS X 10.4 (aka Tiger). It originated under linux (Suse 8.4), and later, after I bought my PowerBook, migrated to MacOS X (no porting was necessary. I just copied it over, and it ran). I have run it to a limited extent under Windows XP, and it seems to work. (However, there are a few places that it "shells out" from `perl`, which might not work under Windows. I also cannot swear that there isn't a missing "`binmode`" somewhere.) But I *have* run POSDA under Windows (really, I have), and even developed some Windows specific scripts. I would like to be able to say that POSDA runs "like a native" under Windows, but I just don't have the heart (stomach?) to use Windows that much. There are other things in life.... So, feedback from Windows users is appreciated.

POSDA also has a database component, which uses `postgresql`. The scripts which access a database have only been tested using `postgresql`, and use the "natural join" syntax a lot (I don't know how portable a "natural join" is). However, `postgresql` installs and runs on just about any UNIX (and Windows, even in native mode), so it meets my needs.

I have run POSDA on both big and little endian systems (the big-endian system being my beloved 12 inch PowerBook G4). I have tested some with big-endian encodings, but by and large most DICOM files are little endian (DICOM is little endian at heart). So take any big endian stuff with a grain of salt.

I use POSDA most often on my laptop, which is a 12" PowerBook G4 with MacOS X version 10.4. I'm running `perl` 5.8.8 on this, but I have tested with the newly released `perl 5.10`, and it seemed to run fine (but I haven't upgraded to 5.10 (and don't really intend to soon). I also fairly regularly run POSDA under Fedora Core 2 running `perl` 8.4. If you get POSDA installed and running on another system, let me know and I'll start a list.

Bottom line: POSDA may help you solve some problems, but it's not intended to be the basis of a commercial grade product. You may have to do a little code reading and trouble shooting (that's why the code's there), and the documentation may be a little sketchy and informal. But it is open source.

POSDA is for the “Casual Programmer”

“Casual Programmer” does not mean “Amateur Programmer” or “Non-Professional Programmer”. By “Casual”, I mean this: Its not designed to professional standards and I wouldn’t use it to implement a system to professional standards. But there are plenty of times you just need to “write a little program to solve a problem”. The program might be a throwaway, or it might evolve into something of lasting value. But it needs to be done quickly, and easily, without the “overall system considerations” getting in the way of the task at hand.

There are some architectural considerations in the implementation of POSDA which arise from this:

- Don’t expend too much effort on efficiency. You (actually I) will just get it wrong anyway. Machines should work. People should lay about thinking deep thoughts and uttering profundities. If you optimize, optimize for ease of use and modification. You’ll (actually I) will be needing to fix it.
- Don’t try (for example in the case of anonymization) to make a grand tool which solves all problems. Just make a toolkit which enables people (i.e. Casual Programmers) to solve their own problems. The problem will be a little different every time it presents itself anyway...

So POSDA is for the “Casual Programmer”, and if you don’t find yourself in the position (at least sometimes) of being a “Casual Programmer” (i.e. you’re willing and able to do a little programming, but don’t want to build a whole system), then perhaps POSDA is not for you, and you should look at some of the other open-source DICOM packages out there.

If you look at the POSDA database schema, you’ll see that POSDA take a pretty causal approach to database schemas also. Same arguments apply. And it may not be for you.

POSDA is DICOM RT oriented

The company which I work for (CMS, which is now part of the Elekta family, which has given gracious consent to releasing POSDA as an open-source package), is in the RT business, and I have been looking at DICOM RT implementation issues with it for a while now, so POSDA has picked up a bit of an RT orientation. It didn’t start out with an RT orientation, but it has acquired one (not by design, but by need). So objects which are important in RT (such as RT Plan, RT Structure Set) are well represented. Objects which aren’t (Digital X-Ray comes to mind) aren’t.

Some examples of the kind of things for which I have used POSDA

I keep a database of all the DICOM images which my company encounters. Sometimes I get questions about what kinds of DICOM data I have. Here's some examples of problems I have addressed with POSDA:

- How many of them are duplicates? (what's that? What do I mean by "duplicate"? Well, OK then)
  - How many of them have the same SOP Instance UID?
  - Which of these are really identical, and which are different? What are the differences?
  - How many of them are the same image (i.e pixel data) with different SOP Instance UID's?
  - How many of them have the same SOP Instance UID, but different pixel data? [Here's the good news: I've really only seen this in one set of images. The bad news? The set of images came from the NEMA website, and was prepared by WG-07. Sigh]
- How well do various vendors do in maintaining consistency in Study Level Information?
- Given a study with RT Structure Sets, RT Dose, RT Plan, and CT's, is the RT Structure Set and RT Dose really positioned correctly (given that I have access to the correct non-DICOM representation of same)?
- Given a study with RT Structure Sets, RT Dose, RT Plan, and CT's, can you anonymize the whole mess leaving all the references intact (changing UID's for Study, Series, and Image also of course)?
- How many of the RT Structure Sets have an ROI which could mean "prostate"? Can you get me the entire study (CT's, RT Struct, RT Plan, RT Dose) which includes such ROI's?
- Can you get me a list of all the RT Plans by manufacturer with the number of beams and total control points, grouped by beam and control point
- Can you get me all the plans with motorized wedges?
- How often do you see DICOM files improperly padding text strings with NULLS?
- What are all the names that you know of that have been used in ROI name?
- Same as above, but for vendor x?
- Do you have any CT's in which the Image Orientation Patient reflects a gantry tilt?

I have also written an "Anonymizer" which can anoymize a set collection of DICOM images while preserving the internal linkages within the images.

I also wrote a "Pseudo Phantom" generator. This is a program which generates a set of "geometrically ideal" CT series (and related PT series). It is capable of generating these series in different frames of references. In fact, doing this precipitated POSDA being finally released as open source.

History of POSDA (in one page)

I first wrote the core set of routines about 6 years ago, in the course of doing a contract for Johns Hopkins. The project involved extracting a bunch of ACR-NEMA 2.0 images from a dead archive, translating them into DICOM 3.0 format, storing them in a RAID, and building a database of information about the images. When I first starting working on the contract (it was a small contract, without much time or money, just after the dot-com bust), I looked at the open source options for DICOM parsers. Pretty quickly, I decided that there was a big learning curve associated with any of the existing tool kits, what I really wanted was a DICOM/ACR-NEMA parser written in `perl`, so I could just write little `perl` scripts to do the work. (Its so much easier if you can use scripting languages to do this kind of stuff). I found a small DICOM parser written by Andrew Crabb (who was incidentally, also at John's Hopkins). It really didn't do all I needed, but it was a start. So in a couple of weeks I had reworked it into a DICOM/ACR-NEMA parser, DICOM 3.0 converter, etc.

As I was doing the contract, and working with the guys at Hopkins, I decided that when I finished, I would put the stuff out as open source, and see if I could get something going around an open source DICOM Archive, written in `perl`. In fact, it was written into my contract with them that all of the software I wrote for the contract would be available under the "same license as `perl`" (i.e. the so called "artistic" license). I came up with the name "POSDA" for Perl Open Source DICOM Archive. Notice that I call this a "DICOM Archive," not a PACS. Its was really just a storage/search device, not a PACS (which includes display systems). It also eschews other PACS baggage like image routers, etc. Not that I feel that these functions are unnecessary, but I think that they aren't part of an archive. A router (in my opinion) is probably better off as a separate device which talks to the archive like everybody else (depending on how "everybody else" talks to the archive).

Anyway, I never got around to setting up an open source project and repository for POSDA (although I did get the domain name posda.com). Instead, I wound up getting a job that (initially) didn't really involve much DICOM. Later, after some changes of fortune at my place of employment, I would up with a lot of DICOM responsibilities again, and had the need to look inside a lot of DICOM files. What I needed was a set of `perl` routines which would allow me to write little `perl` scripts.... So I resurrected POSDA, and set out using it.

It worked pretty well for me. But I was shocked (absolutely shocked!) at just how ugly this darn parser was. But it basically worked, and I could fix the bugs pretty fast when I found a file that wouldn't parse, so after a while, I had what I think is a pretty solid parser. But it was still ugly (this may have had something to do with writing it in two weeks, and quickly fixing bugs as they popped up for a few of years). I just couldn't bring myself to let anyone else take too close a look at it. Too ugly.

Finally, I couldn't take it anymore, and decided to refactor it. Surprisingly, it took less than a day to refactor (and it passed regression tests on the roughly 100,000 files which I had accumulated for test in less than a week). Its still pretty ugly, but not so embarrassingly ugly that I can't share it. And ugliness (like beauty) is really just in the eye of the beholder anyway.

Along the way, I wrote a little package to anonymize DICOM files while preserving the linkages between dose, plans, and structure set. And a little program to fix up linkages between structure sets and dose. The anonymizer was actually shared with a few select users who were supplying us with anonymized data sets. But it was when I wrote the "Pseudo Phantom" generator at the IHE RO Domain testing and Technical Committee meeting in Munich that I got a request from outside to release these scripts to the DICOM community. So I sent the request up the chain of command, and got approval.

Then I had to write some documentation.... Yikes! Open Source may be harder than it looks.

This POSDA Distribution (version 0.5)

POSDA is distributed as a single `tar` file. When expanded, this tar file expands into the following directories:

- `bin` -This directory includes a number of perl programs which use POSDA.
- `include` -This directory contains the perl modules which comprise POSDA.
- `config` - This directory contains configuration files. At the present time, all of the configuration files are configuration files for the Pseudo Phantom Generator. This may change in the future.
- `psv` - This directory contains definitions for various types of data dictionaries in "Pipe Separated Text" (hence "psv") format.
- `sql` - This directory contains a number of files which define various database schemas used by POSDA. One of the files also contains a `pgsql` dump of the database for the DICOM data dictionaries.
- `doc` - This is where the documentation goes. It contains this very document (and others).

Posda License

 I did start out by modifying some code by Andrew Crabb. I don't think any of it is left, but he's a good guy, has a good website (www.idoimaging.com), and deserves some credit.

Like so much else in the world of perl, POSDA is licensed using the "Artistic License", i.e. the same license as perl itself. Most perl packages just say "you are free to use this under the same terms as perl itself". I am including a copy (printed as a `pdf` from the `perl.com` website) of the Artistic License with this distribution.

Support

I'm not naïve enough to think I can get by without doing some support. To this end, I have set up a gmail account for POSDA issues: `posda.admin@gmail.com`. However, don't trouble me with trivial issues. Don't try to convince me I should have written POSDA in Python, or Java, or Occam, or whatever. I don't do religious debates. If you have a particular success story with POSDA though, I'd like to know. If you find a bug, I'd like to know. If you'd like to take over Windows support, I'd like to know. I'll try to monitor the email account and get back to you, but I do have a real job (and a life). So we'll see.

Further Reading

From here, you should go to the doc directory, and start reading. The documents contained in the doc directory for this release (0.5) of POSDA are the following:

- `PosdaOverview.pdf` - This document (hope you liked it).
- `ArtisticLicense.pdf` - The Artistic License printed out from the perl.com website
- `PosdaInstallation.pdf` - How to get and install POSDA. You have probably already read this document to get here, but if not, here it is.
- `PosdaPseudoPhantoms.pdf` - How to use POSDA to generate geometrically ideal "Pseudo Phantoms" for DICOM testing. This is what has precipitated this release, so it makes the cut for the first release.

Various other documents are in various states of development and will be included in subsequent releases:

- `PosdaAnonymizer.pdf` - How to anonymize DICOM data sets using POSDA. Preliminary versions of this document have actually been released to selected CMS customers, and the Anonymizer has been used successfully outside CMS. It needs a little cleaning up.
- `PosdaUtiilties.pdf` - A list of the programs in the Posda/bin directory, with at least a one liner for each, and a more complete description for selected (i.e. useful) programs.
- `DatabaseFlavor.pdf` - A document which is intended to give you a flavor of how the POSDA database can be used. Not a reference document, just some examples for self starters.
- `PosdaFlavor.pdf` - A document which is intended to give you a flavor of how to write POSDA scripts to analyze DICOM files. Not a reference document, just some examples for self starters.
- `PosdaReleaseNotice.pdf` - A document which contains information about the release. The version number, a list of all the components, and differences between this and earlier versions. I'm not going to go to the trouble of generating this until the second release. I'm in a hurry to get the first release out (some people are waiting).