Development of a Next-Generation Automated DICOM Processing System in a PACS-Less Research Environment

Scott E Ziegler

Published online: 1 May 2012
© Society for Imaging Informatics in Medicine 2012

Abstract The use of clinical imaging modalities within the pharmaceutical research space provides value and challenges. Typical clinical settings will utilize a Picture Archive and Communication System (PACS) to transmit and manage Digital Imaging and Communications in Medicine (DICOM) images generated by clinical imaging systems. However, a PACS is complex and provides many features that are not required within a research setting, making it difficult to generate a business case and determine the return on investment. We have developed a next-generation DICOM processing system using open-source software, commodity server hardware such as Apple Xserve®, high-performance network-attached storage (NAS), and in-house-developed preprocessing programs. DICOM-transmitted files are arranged in a flat file folder hierarchy easily accessible via our downstream analysis tools and a standard file browser. This next-generation system had a minimal construction cost due to the reuse of all the components from our first-generation system with the addition of a second server for a few thousand dollars. Performance metrics were gathered and the system was found to be highly scalable, performed significantly better than the first-generation system, is modular, has satisfactory image integrity, and is easier to maintain than the first-generation system. The resulting system is also portable across platforms and utilizes minimal hardware resources, allowing for easier upgrades and migration to smaller form factors at the hardware end-of-life. This system has been in production successfully for 8 months and services five clinical instruments and three pre-clinical instruments. This system

has provided us with the necessary DICOM C-Store functionality, eliminating the need for a clinical PACS for day-to-day image processing.

Keywords PACS · DCMTK · DICOM · DICOM workflow · DICOM storage

Introduction

The use of medical imaging modalities within the pharmaceutical discovery space provides many benefits which help reduce the discovery cycle time and overall cost [1]. Although we do not deal directly with patients and clinical diagnostics, we still face some of the same challenges found in the clinic related to image data processing workflows and image management. The primary interest in the discovery setting is quantitative medical imaging, which translates to data sets larger than typical clinical data sets. For example, when using a multi-detector CT, the data of interest are the thin-sliced data rather than the thick-sliced data, which, in a clinical setting, are typically sent to the Picture Archiving and Communications System (PACS) for long-term storage [2].

Typical clinical PACS provides many features that are not required within the pre-clinical research setting such as teleradiology capability, radiology workflow management, query and retrieve capability, and interfaces for external systems such as Hospital Information Systems, Electronic Medical Records, and Radiology Information Systems. A clinical PACS, including hardware, software, support contracts, and local support, can cost millions of dollars, making it difficult to build a business case and to calculate a return on investment within a pre-clinical research environment. Our institution's clinical imaging systems (herein referred to as

S. E. Ziegler (✉)
Merck & Co.,
770 Sumneytown Pike,
West Point, PA 19486, USA
e-mail: scott_ziegler@merck.com

instruments), and several of our pre-clinical imaging systems, produce images in the Digital Imaging and Communications in Medicine (DICOM) standard file format. Our first-generation DICOM server system was nearly identical in concept to the system discussed in Yakami et al. [2] using the Free Open Source Software (FOSS) OsiriX software system and a flat file folder hierarchy housed on a high-performance Network Attached Storage (NAS) appliance. This allowed the operator, at the end of an acquisition, to DICOM transfer the study to the storage system for analysis, just as one would in a clinical setting employing a PACS.

The DICOM standard specifies that an application which uses the DICOM protocol is called an application entity (AE), and an AE within a DICOM network is identified by three parameters: <IP Address, AE Title (AET), and TCP Port>. This allows one hardware server to host many DICOM Service Class Providers (SCPs), or servers, by specifying different AETs and port numbers. When two AEs need to communicate, one acts as a Service Class User (SCU), or client, and the other as the SCP. When the SCU establishes a connection with the SCP, it is called an association. The SCU sends its data to the SCP and then the SCU terminates the association.

Our first-generation system was built using the OsiriX software system to function as the DICOM storage SCP, running on a single Xserve® (Apple Inc.) and writing all of the incoming DICOM files into a single directory. The preprocessing program performed a first-pass sort of these files into individual instrument folders by reading each DICOM file. Upon completion of the sort, each new resulting instrument folder was processed in a serial manner. While these instruments were being processed, if another study was sent to the server, the processing of the new study would have to wait until the first data set was complete. OsiriX utilizes a relational database (RDB) just like typical DICOM servers and PACS. These RDBs are typically populated with key DICOM tag metadata extracted from the received files. A typical use of an RDB enables DICOM query capability for users to search and retrieve DICOM files from the server or PACS. For our purposes, data flow is unidirectional from the instrument to the storage and does not need to be queried and retrieved back to the original instrument, eliminating the need for the RDB. The OsiriX database in this implementation periodically required user intervention to purge processed studies.

This first-generation system works well, but the preprocessing program, referred to as the arranger in Yakami et al. [2], became difficult to maintain and did not scale well as additional instruments and modalities were added. This preprocessing program was initially designed for processing a single instrument, but over time, it evolved into processing five instruments, some with multiple modalities. This single piece of preprocessing code used several very large

conditional statements, in turn making this program very difficult to maintain. The original preprocessing code was written in MATLAB® (MathWorks USA) and was executed within the MATLAB® environment by interpreting the source code. Running in this environment also made MATLAB® upgrades difficult to perform without impacting the production system. Additionally, no hardware redundancy existed, making the server a single point of failure for our DICOM workflows. With three additional DICOM sources now needing to transmit files, study volumes increasing, and some multi-modality studies generating in excess of 7,000 files per study, a more efficient and maintainable way to process our DICOM data was required.

A next-generation, modular automated DICOM processing system (ADPS) has been developed that, from the end user perspective, operates the same as the first-generation system, eliminating any user training or learning curve. This new system is composed of five core components:

1. The corporate network
2. A high-performance NAS system
3. Dual Xserve® servers, providing system redundancy and an environment for both development and testing
4. Instrument-specific MATLAB®-based image preprocessing components
5. The FOSS DCMTK version 3.6.0 (OFFIS E. V., Oldenburg, Germany) [3] library

Materials and Methods

The core components of the ADPS can be broken down further into sub-components and configurations. The entire system is connected via our corporate Ethernet-based local area network (LAN). Our network connectivity is a 1-Gb/s network. Our DICOM servers and storage are co-located on the same site, with 1-Gb/s end-to-end connectivity, to reduce file transfer latency. The network switch that connects to the NAS storage system is configured with IEEE 802.1AX-2008 link aggregate protocol [4] aggregating four 1-Gb/s network lines into a single network connection. This configuration provides minimal average ping latency from the client of 0.4 ms for a 64-byte packet.

The NAS system is a Bluearc (Bluearc Corp., USA) Titan system utilizing a fiber channel high-speed redundant array of inexpensive disks (RAID) [5]. The system is configured for RAID level 5, IEEE 802.1AX-2008 network interface, and redundant Titan controllers. The Bluearc system currently houses 145 TB of disk space and was originally selected for its performance specifications and its theoretical single-file share size of 256 TB. File shares are exported using both Network File System (NFS) and Common

Internet File System (CIFS) protocols. This system is mirrored nightly to a tier 2 storage system located at another site.

Our standard folder structure to organize the DICOM and preprocessed data has been designed as shown in Fig. 1. This structure organizes the data by project name, instrument, study name, and image type. The information entered into the patient name <0010,0010>, and PatientID <0010,0020> tags at the instrument have been standardized, and the contents of these tags are utilized to create the folder hierarchy. The DICOM image folder name is created using a prefix from the modality tag <0008,0060>.

The NAS system is configured using access control list (ACL) policies and security so that the original data are restricted to those who require access. All users within the group are granted read-only access, with write access being granted on an as-needed basis. All of the image files are accessible via CIFS and NFS protocols with the same permission constraints.

The server hardware utilized in this system consists of two Apple Xserve® systems co-located within the same subnet as the storage system. The production server is configured with an Intel 2.26-GHz quad-core processor, 12 GB of RAM, and is running OS X Server 10.6.7 with a single 160-GB Serial Attached SCSI (SAS) hard disk. The backup server, which is also used for development, is configured with a 2.26-GHz quad-core processor, 12 GB of RAM, OS X Server 10.6.7, and two 1-TB SAS drives, with one used for backups using Apple's Time Machine®. The local hard disks are also used for temporary storage of the incoming DICOM images. The production server is configured with the MATLAB® Runtime Engine, whereas the backup/development server has the MATLAB® development environment installed.

The image preprocessing components are written in MATLAB® 2010a and compiled to run on OS X 10.6. These preprocessing components are specific for each instrument and modality. The system is constructed so these

components can easily be plugged in without affecting any other instrument workflow. These preprocessing programs perform the following functions:

1. Create the appropriate study folder on the storage system
2. Create the DICOM and analyze folders under the study folder
3. Move the DICOM files from the server's temporary storage to the target folder
4. Clean up the local storage
5. Preprocess the DICOM images by converting to the analyze 7.5 format [6] for use with our internal analysis tool suite
6. Send e-mail notifications to the appropriate users that the DICOM and Analyze images are ready including links to the data

When compiling MATLAB® code for the OS X target platform, the compiler, along with creating the executable, also generates a shell script that is used to set up the MATLAB® Runtime Environment and launch the preprocessing program. This shell script has been modified for our specific paths and used as a template when adding new instruments to the system.

The image preprocessing component varies depending upon the modality and particular instrument. The preprocessing component performs the conversion from DICOM to Analyze and performs any necessary steps such as unit conversions, decay correction, scaling, and image orientation changes. The preprocessing component uses XML-based configuration files, so when something changes, such as the e-mail notification list or any other parameters specific to the instrument, the program does not need to be recompiled.

The server software, which acts as the SCP and receives the DICOM images from the instruments, is the DCMTK library storescp [7] DICOM C-Store SCP program. This

Fig. 1 Folder structure used to store DICOM and Analyze 7.5 images. The project name is extracted from the Patient ID tag <0010,0020>, the instrument name is known, the study folder name is created from information entered into the patient name tag <0010,0010>, and the DICOM folder name uses a prefix determined by the DICOM modality tag <0008,0060>

software is free open-source software, is robust and reliable, used throughout the Imaging Informatics community [8–12], and can be downloaded and compiled for most target platforms. The storescp program is rich with command line configuration options, which define the SCP behavior. For this system, we have utilized the command line options shown in Table 1.

The storescp xcs option works in conjunction with the tos option, which sets the time in seconds for concluding when the study has completed being transmitted. This option can be used to help in networks where there may be high collision rates or long latencies. At the end of the timeout period, the program that is referenced by the xcs option is executed. The xcs option also has optional parameters associated with it that provide access to various data related to the transfer. In our configuration, we are using the #p parameter which contains the absolute path to the folder where the incoming DICOM files were written. This parameter is then passed as a command line argument to an application shell script, written in the Bourne Again Shell (BASH) scripting language.

The production server system is configured to automatically launch the storescp listeners when the system is rebooted and the user is logged in. It was preferred to configure these processes to run at the user level due to the permission and security constraints imposed by the storage system. We found it more secure to restrict read/write access to a user account rather than a system-level user such as root. The servers are located with the NAS in a secure data center without any local monitor/keyboard or mouse and are only accessed remotely via a username and password.

The backup/development server does not start the listeners automatically, and they would only be launched in the event the production server is unavailable. The instruments have been configured with both DICOM locations and the production server as the default. To use the backup, the operator

simply makes a different selection from the list of DICOM locations when sending the study.

BASH scripts configure the DCMTEK environment and launch each storescp instance as a background process, with one instance running per instrument. The BASH scripts were written using environment variables, so when a new instrument is integrated into the system, the script could easily be duplicated by copying an existing script and changing the variables. An example listener startup script is shown in Fig. 2.

The OS X launchd [13] system is used to launch the listener programs at user login. A user-level XML plist configuration file, generated with the Lingon [14] application, is located in the \$HOME/Library/LaunchAgents folder and is read by the launchd process at user login. This launch agent calls a top-level shell script which then launches the individual listener scripts. An example of the startup plist and top-level startup script is shown in Fig. 3.

The advantage to this configuration is that additional listeners may be integrated into the system without having to modify the existing startup scripts. A new listener shell script is simply added to the listener folder.

Testing was performed by utilizing a sample study data set consisting of 3,256 PET slices and 737 associated CT slices, for a total of 1,133 MB of data. A Mac Pro host with 16 Gb of RAM and dual 3.2-GHz quad-core Intel Xeon processors was configured with the DCMTEK C-Store SCU program (storescu) known as the client. The storescu program establishes an association with a SCP and performs a DICOM transfer just as if a user was sending studies from a DICOM-enabled instrument. The storescu command line interface utilized three arguments: the SCP host name, the SCP listening port, and a list of DICOM files passed as *.dcm. Ten SCPs were configured on the DICOM server and ten SCUs on the Mac Pro. The network interface between the client and the server consisted of a 1-Gb/s Ethernet LAN with each host residing in its own subnet, on the same physical site, and with a single router connecting the subnets. A simple top-level script was written to enable the simultaneous launching of the SCUs.

Table 1 List of storescp options used in this system and their definitions

Option	Description
-od <directory>	Defines the output directory where the incoming DICOM files will be written
-aet <name>	Application Entity Title
-tos <seconds>	Time to wait before concluding transmission is complete
-rms	Renames the output files according to a certain pattern at the conclusion of the transfer
-ss <prefix>	Sort studies based on prefix
-xcs <program>	Execute on end-of-study
-v	Verbose mode, used for log file capture

Results

Three primary objectives motivated the development of this second-generation system:

1. Improve system maintainability and uptime
2. Improve the processing performance
3. Minimize construction costs

The results were measured against these objectives, the first-generation system, and include metrics related to processing speed and image integrity.

Fig. 2 Example storescp listener startup script

```
#!/bin/bash
export DCMTKROOT=/Users/analysis/dcmtdk-3.6.0
export DCMDCPATH=$DCMTKROOT/dcmdata/data/dicom.dic
APP=$DCMTKROOT/dcmnet/apps/storescp

AE_TITLE=TRIO_DICOM
S_SORT=TRIO
PORT=50006
OUT_DIR=$HOME/DICOM_XFER/TRIO
LOGFILE=$HOME/Logs/TRIO/trio_dicom.log

$APP -od $OUT_DIR -v -aet $AE_TITLE -tos 5 -rns -ss $S_SORT $PORT -xcs
"$HOME/bin/run_procTRIOStudy.sh #p" 2>&1 > $LOGFILE &
```

Improve System Maintainability and Uptime

This primary objective was met in several ways; the first was by segmenting the all-in-one preprocessing program into instrument-specific programs. This allows for the development and testing of individual instrument preprocessing programs without impacting any of the preprocessing programs for the other instrument workflows. This modularity has eliminated a very large, difficult to maintain, complex piece of code.

The second was by the use of individual DICOM C-Store SCP listeners, one for each instrument, combined with the top-level looping script to launch them. The addition of listeners allows new instrument workflows to be added without impacting any existing instrument workflows. Individual listeners can also be disabled without impacting any other instrument workflows.

The third was by the addition of a second server. The first-generation system utilized a single Apple Xserve® and a single OsiriX DICOM listener. The additional server provided us with necessary redundancy to maintain uptime, and the original server was repurposed as a development/test environment and a backup listener server. This eliminated our single point of failure and the ability to render the entire processing system inoperable by simple code changes or a

shutdown of OsiriX. Each DICOM-based instrument has been configured with the production server as the default DICOM location, and the backup server is configured as a DICOM location in the transfer menu. This also provided us with the ability to perform MATLAB® upgrades and test them without impacting the production system. The production server uses the MATLAB® Runtime Engine, eliminating the need for additional MATLAB® licenses and installations.

Improve the Processing Performance

This second-generation system utilizes a DICOM C-Store SCP for each instrument SCU and an associated instrument-specific preprocessing program. The SCP listens for SCU requests and establishes an association. Upon completion of an association and expiration of the SCP tos, the DICOM SCP is available for the next association even while the preprocessing program is running. The SCP is configured to sort the files into individual study folders as they arrive, thus eliminating the first level, time-consuming sort done previously. The storescp program, at the end of the time out period, launches the preprocessing program. For a study consisting of a single modality and 3,256 files, the transfer and sorting time was reduced from 2 min and 42 s on the

Fig. 3 Example plist startup script and top-level listener launch shell script

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.merck.launchListeners</string>
  <key>ProgramArguments</key>
  <array>
    <string>/Users/analysis/bin/launchListeners.sh</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
</dict>
</plist>

#!/bin/bash

L=`ls -l $HOME/bin/listeners`
for script in $L
do
  $HOME/bin/listeners/$script
done
```


first-generation system and to 30 s on the new system. The original system also had utilized a UNIX cron job that ran every 4 min to check for new files and start the sort and preprocess program. Overall, the new system eliminated this 4-min wait and reduced the sorting by 2.25 min for a total savings of over 6 min from this example scenario.

The preprocessing programs are compiled utilizing the MATLAB® Compiler™ toolbox and executed by the MATLAB® Runtime Environment. This configuration eliminates the overhead resources (CPU, memory, etc.) and licenses required to launch multiple instances of the MATLAB® environment and eliminates the time MATLAB® requires to interpret the code for execution.

This second-generation system will, at a minimum, process the same volume of studies as our first-generation system, but with the utilization of multiple SCPs, it can now process studies in parallel rather than serially. Due to this change, performance, load testing, scalability, and integrity tests were performed to ensure that the hardware could handle the load of our existing instruments, and future instruments, should all instruments send studies simultaneously.

Minimize Construction Costs

This second-generation system was constructed around the original NAS system and folder hierarchy, which eliminated the need to restructure existing data or purchase additional disk space. Additional savings were realized by: utilizing the existing corporate network, repurposing the original Xserve®, reusing the existing MATLAB® licenses, and utilizing the FOSS DCMTK. We purchased an additional Xserve® in July 2010 with the AppleCare® protection to be used as our production server and to provide hardware redundancy.

Performance Metrics

An initial baseline was established by using the client to send a single copy of the study to the server. The total average ($N=5$) measured transfer time was 26 s, with a variance of 2.3 s. Ten simultaneous studies were sent from the client, and the total transfer time was measured at 124 s. Server metrics were monitored during the transfer of the ten studies using the OS X Activity Monitor and recorded. The peak network utilization was 101 MB/s, the maximum memory usage was 3.4 MB per storescp instance, and the minimal CPU usage per storescp instance was 15 %.

Additional analysis was performed to determine the scalability of the server by measuring and plotting the DICOM association times for data transmission for an increasing number of simultaneous transmissions. The initial test transferred one study and was then incrementally increased to ten simultaneous study transmissions. The results are shown in Fig. 4.

The plot in Fig. 4 indicates that the system, while sharing server resources such as the network interface, hard disk, CPU, etc., behaves in a linear fashion, and as a result, the DICOM transfer portion of the system is highly scalable, allowing for significant future growth. The network utilization of the SCP was also monitored during the transmission of the ten simultaneous studies using the built-in OS X network utility. A peak utilization of 110 Mb/s was observed out of the theoretical peak of 1,000 Mb/s.

Benchmarks were conducted using representative data sets from four clinical DICOM instruments to account for the preprocessing module of the system. These preprocessing programs create the target study folders, move the transferred DICOM files to the proper target study folder, perform conversion of the DICOM data set to the Analyze 7.5 file format, and apply any scaling, unit conversions, etc. As a result, the resources required for these processes are highly instrument- and modality-dependent. Table 2 outlines the study-specific data sizes used for the test.

Baseline times were established by transferring and preprocessing each representative study independently, five times, and the average transfer time computed ($N=5$). The four representative studies were then transmitted and preprocessed simultaneously, and the time to completion for each study was measured and compared to the corresponding baseline time and recorded as a delta. Table 3 contains the times and the delta.

The delta values in Table 3 indicate that when processing multiple simultaneous studies, the additional contention for system resources has a negligible effect on the overall processing time, with the largest study taking only four additional minutes.

DICOM Image Integrity

The integrity of the DICOM images transferred via the DCMTK storescp program was evaluated by comparing the storescp-received files to the same files received via our first-generation system. MATLAB®, along with the dicomread and isequal functions, was used to process each file transferred, and we found 0 % error out of a 3,256 DICOM slice sample data set.

Discussion

This second-generation DICOM processing system has been in service successfully for 6 months. The simple implementation using the DCMTK library makes it highly desirable in non-clinical environments when a traditional PACS is cost-prohibitive to purchase, install, or maintain. The storage system and folder hierarchy has been used successfully since 2003, and the DICOM preprocessing methodologies,

Fig. 4 Plot of the number of storescp server instances and simultaneous transfers vs. time to receive the studies. All transfers utilized the same study set

although updated occasionally, have been used successfully for several years.

The capability of the storescp program to launch another program at the conclusion of a DICOM transfer allows us to run distinct instances of the SCP for each instrument. This feature also enabled the creation of preprocessing programs for each instrument, thus creating a modular system. The modularity of the system makes for easier maintenance as a single instrument workflow can be maintained without affecting other workflows and the system is broken down into smaller more manageable components.

The use of the DCMTK library also provides a level of portability as binaries are available for the major operating system platforms as well as the source code for compilation on any other platforms. This will be beneficial to us when our servers reach their end-of-life cycle as Apple has discontinued production of the Xserve® hardware.

The new system has significantly improved overall system uptime by having individual SCPs rather than a single OsiriX SCP. The individual preprocessing programs have eliminated the single preprocessing program, which would take the entire system down with the simplest edit or error and for the duration of the integration of a new instrument. A redundant server has also provided a development and test environment, which eliminates the impact on the production environment when things change or new instruments are integrated.

The use of FOSS and common off-the-shelf hardware components makes the system simple to maintain and

simplifies system upgrades when hardware reaches its end-of-life.

Conclusion

This paper has presented a next-generation, high-performance, high-capacity, and highly scalable DICOM processing system that was successfully designed and implemented at minimal cost. The second-generation system was built using dual hardware servers, the free open-source software DCMTK library, and compiled MATLAB® preprocessing programs. DICOM image integrity has been verified and system performance and scalability have been tested and found to be highly satisfactory. The system is fully capable of processing multiple simultaneous DICOM transfers from multiple instruments and modalities with minimal resource utilization. This configuration provides for simplified hardware migrations when the current hardware reaches its end-of-life. The use of a single plist/shell startup script, a standard SCP shell script, XML-based configuration files, elimination of an RDB, elimination of the first-level file sort, and individual instrument/modality preprocessing programs has significantly improved overall system maintainability.

Table 2 Instrument-specific study sizes used for performance testing

Instrument	Total files	Size (MB)
PET/CT 1	3,993	1,133
PET/CT 2	370	123
SPECT/CT	158	116
MRI	981	143

Table 3 Comparison of the baseline individual transfer times to the simultaneous processing times

Instrument	Baseline	Time to completion	Delta
PET/CT 1	0:09:28	0:13:34	0:04:05
PET/CT 2	0:01:02	0:02:11	0:01:09
SPECT/CT	0:00:32	0:00:44	0:00:11
MRI	0:01:52	0:02:17	0:00:25

The delta is the time consumed by the simultaneous execution of the preprocessing programs relative to the baseline

References

1. Hargreaves R: Molecular Imaging in Drug Discovery and Development. Available at <http://www.molecularimagingcenter.org/index.cfm?PageID=9130>. Last accessed 20 May 2011
2. Yakami M, et al: Development and evaluation of a low-cost and high-capacity DICOM image data storage system for research. *J Digit Imaging* 24(2):190–195, 2011
3. DCMTK library. Available at <http://www.dcmk.org/dcmk.php.en>. Accessed 20 May 2011
4. IEEE 802.1AX-2008 Protocol. Description available at <http://en.wikipedia.org/wiki/802.3ad>. Accessed 20 May 2011
5. Dandu RV: Storage media for computers in radiology. *Indian Journal of Radiology and Imaging* 18(4):287–289, 2008
6. Analyze 7.5 image format. Standard available at <http://eeg.sourceforge.net/ANALYZE75.pdf>. Accessed 20 May 2011
7. Storescp description at <http://support.dcmk.org/docs/storescp.html>. Accessed 20 May 2011
8. Nagy P: Open source in imaging informatics. *J Digit Imaging* 20 (Suppl 1):1–10, 2007
9. Gatta R, et al: Open-source, low-cost, high-reliability solutions for digital imaging systems: example of a dicom router. *La Radiologia Medica* 112(8):1252–1259, 2007
10. Trevisan DG et al.: A medical component-based framework for image guided surgery. In: *Multimodal User Interfaces from Signals to Interaction*. Berlin: Springer, 2008, pp 271–280
11. Channin D, et al: The caBIG™ Annotation and Image Markup Project. *J Digit Imaging* 23(2):217–225, 2010
12. Wolf I: Toolkits and software for developing biomedical image processing and analysis applications. *Biomedical Image Processing*, 2011, pp 521–544
13. Launchd. Description at <http://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man8/launchd.8.html>. Accessed 20 May 2011
14. Lingon. Available at <http://www.macupdate.com/app/mac/19879/lingon>. Accessed 20 May 2011