Journal of Digital Imaging

# The Image-Guided Surgery Toolkit IGSTK: An Open Source C++ Software Toolkit

Andinet Enquobahrie,[1] Patrick Cheng,[2] Kevin Gary,[3] Luis Ibanez,[1] David Gobbi,[4] Frank Lindseth,[5] Ziv Yaniv,[2] Stephen Aylward,[1] Julien Jomier,[1] and Kevin Cleary[2]

**This paper presents an overview of the image-guided surgery toolkit (IGSTK). IGSTK is an open source C++ software library that provides the basic components needed to develop image-guided surgery applications. It is intended for fast prototyping and development of image-guided surgery applications. The toolkit was developed through a collaboration between academic and industry partners. Because IGSTK was designed for safety-critical applications, the development team has adopted lightweight software processes that emphasizes safety and robustness while, at the same time, supporting geographically separated developers. A software process that is philosophically similar to agile software methods was adopted emphasizing iterative, incremental, and test-driven development principles. The guiding principle in the architecture design of IGSTK is patient safety. The IGSTK team implemented a component-based architecture and used state machine software design methodologies to improve the reliability and safety of the components. Every IGSTK component has a well-defined set of features that are governed by state machines. The state machine ensures that the component is always in a valid state and that all state transitions are valid and meaningful. Realizing that the continued success and viability of an open source toolkit depends on a strong user community, the IGSTK team is following several key strategies to build an active user community. These include maintaining a users and developers' mailing list, providing documentation (application programming interface reference document and book), presenting demonstration applications, and delivering tutorial sessions at relevant scientific conferences.**

**KEY WORDS: Image-guided surgery, open source, visualization, registration, tracking and agile software development**

## INTRODUCTION

Minimally invasive procedures are becoming increasingly popular in today's healthcare system. Patients prefer these procedures to open surgeries because they cause less trauma to the body and result in faster recovery times. Interventional radiologists and surgeons are also becoming more experienced and comfortable performing these procedures. Interventional radiologists use instruments such as needles and catheters to perform diagnostic and therapeutic procedures guided by images. Examples of diagnostic procedures include biopsy of suspicious lesions and injection of contrast agents for computed tomography (CT) angiography. Therapeutic procedures include using stents to open clogged arteries and radiofrequency ablation techniques to destroy tumor tissues.

In image-guided surgery procedures, the surgical instruments are placed through incisions guided by preoperative images. To track the position of the surgical instruments, a tracking system is used. The tracking system shows the position and orientation of the surgical instrument

*[1]From Kitware Inc., Clifton Park, NY, 12065, USA.*
*[2]From the Imaging Science and Information Systems (ISIS) Center, Department of Radiology, Georgetown University Medical Center, Washington, DC, 20007, USA.*
*[3]From the Division of Computing Studies, Arizona State University, Mesa, AZ, 85212, USA.*
*[4]From Atamai Inc., London, ON, N6B 2R4, Canada.*
*[5]From the SINTEF Health Research and the National Center for 3D Ultrasound in Surgery, Trondheim, Norway.*
*Correspondence to: Andinet Enquobahrie, Kitware Inc., Clifton Park, NY, 12065, USA; Tel: +1-1518-3713971; e-mail: andinet.enqu@kitware.com*

*Online publication 17 August 2007*
*doi: 10.1007/s10278-007-9054-3*

in the context of preoperative images. The preoperative image is aligned with the patient coordinate system using a technique called registration. The tracking system is used to track the position of the surgical instrument. A typical image-guided system is shown in Figure 1.

The use of open source software for medical procedures like image-guided surgeries is gaining more acceptances in recent years. With US government agencies such as the National Institutes of Health (NIH) and the National Science Foundation encouraging open source software development, more research groups are developing and releasing software as open source. The Visualization Toolkit (VTK) and the Insight Toolkit (ITK) are prime examples of the success of open source software projects. VTK[1], originally developed at GE Global Research and now supported by Kitware, provides a wide range of algorithms in computer graphics, image processing, and visualization. The toolkit has a wide user base with thousands of users worldwide. VTK provides advanced multidimensional visualization algorithms and modeling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, Delaunay triangulation, and parallel computing.

ITK[2], developed under the support of the US National Library of Medicine (NLM) at the NIH, provides advanced registration and segmentation algorithms. The toolkit contains state-of-the-art algorithms and continues to be updated and maintained with supports from NLM. Similar to VTK, the software is implemented in the C++ programming language and provides wrappers for Tcl, Python, and Java interpreters.

The composition of the ITK development team demonstrates the strength of a collaborative effort that is greatly emphasized in an open source project. The initial development team consisted of three commercial partners (GE Corporate R&D, Kitware, and MathSoft) and three academic partners (University of Tennessee, University of North Carolina, and University of Pennsylvania). Other developers have since joined the effort, and to date, over 50 developers have directly contributed code to the toolkit.

Several benefits have contributed to the increased popularity and acceptance of open source software. Some of the main benefits are listed below.

1) Open source software encourages collaboration between academic, commercial, and government institutions by providing a common software base and creating a sense of community.
2) Open source software saves resources by avoiding "reinventing the wheel." Oftentimes, a researcher wastes valuable time and resources implementing basic infrastructure. Using open source implementations of basic functionality, researchers are able to focus on new research efforts.
3) Open source software provides a valuable resource for educational purposes. The best way to learn software development is by studying how other developers have done it and by attempting to improve it.

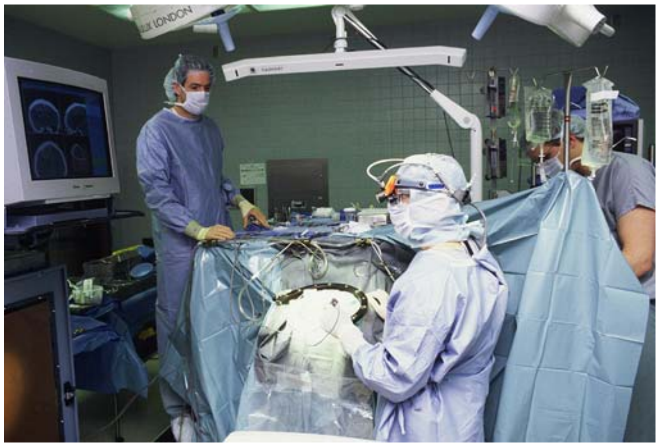


**Fig 1. Image-guided system for brain surgery showing the optical tracking system at the *top right* and the display overlay on the *far left*. The patient is under the blue cover in the *middle*. (Photograph courtesy of Richard Bucholz, MD, St. Louis University).**

4) Open source software provides implementation of reference or benchmark algorithms for validation and verification purposes.
5) Open source software, when distributed under the appropriate license, permits users to try out the technology without restrictions, and this ultimately encourages rapid dissemination of the technology and the growth of a user community.

With the above benefits, open source toolkits are making the transition into mission-critical applications. Forrester's study[3] has shown that companies are rapidly expanding their use of open source software from simple applications (such as email) to mission critical applications (such as customer contact applications). In the medical field, software applications developed using open source toolkits have been submitted to the Food and Drug Administration (FDA) for approval in clinical studies. Although the FDA does not have any specific policy on open source software, the FDA requires that software included in medical devices should be developed following a process for which a quality control system is in place. As one example, the FDA recently approved a single center clinical trial for an electromagnetically tracked lung biopsy application developed using image-guided surgery toolkit (IGSTK) that will begin shortly at Georgetown University Medical Center.

IGSTK, the focus of this paper, is an open source software project developed with support from the National Institute of Biomedical Imaging and Bioengineering (NIBIB) at NIH. It is a cross-platform C++ library that provides the basic components necessary to develop an image-guided system. It is intended for fast prototyping and development of robust image-guided applications. The initial version of the software was released in February 2006 at the SPIE Medical Imaging Conference in San Diego, and an updated version was released at the same meeting in February 2007. The software and documentation is freely available for download at http://www.igstk.org.

The toolkit was developed through a collaboration between academic and industry partners. The principal investigator of the project is Kevin Cleary at the Imaging Science and Information Systems (ISIS) Center in Georgetown University. ISIS is a medical research and development institute that specializes in information and imaging technology in healthcare. The commercial partner is Kitware, a small company specializing in open source software. Kitware also provides commercial consulting services for ITK and VTK. Expertise in tracking systems was provided by Atamai (Ontario, Canada). Experts from Arizona State University provide guidance in software process management and object-oriented technologies. The image-guided research group at Selskapet for Industriell og Teknisk Forskning ved Norges Tekniske Hoegskole (SINTEF) in Trondheim, Norway, has also recently joined the project to provide a strong focus on end-user applications.

The remainder of the paper consists of five parts. "IMAGE-GUIDED MEDICAL PROCEDURES—BACKGROUND" presents background information about image-guided medical procedures. "IGSTK SOFTWARE PROCESS" describes the software process adopted for IGSTK development. This is followed by a discussion of the architecture in "IGSTK ARCHITECTURE." "BUILDING AN IGSTK COMMUNITY" outlines the strategies being followed to build the IGSTK user community. Finally, conclusions are presented in "CONCLUSION."

## IMAGE-GUIDED MEDICAL PROCEDURES—BACKGROUND

Image-guided procedures are rapidly replacing open surgical procedures in clinical practice. The main driving forces behind the wide acceptance of image-guided procedures are technological advancements in medical imaging, registration algorithms, visualization technologies, and tracking systems.

To use medical images for surgical navigation, we first need to make them correspond with the patient's anatomy in a step called *registration*. Registration is the technique of computing a spatial transform that maps points from one coordinate system to another. When we can accurately register the image coordinate system with the patient coordinate system, we can use the images for virtual guidance of surgical instruments.

Registration categories include registration of preoperative images to intraoperative images, registration of preoperative images to the patient coordinate system, and registration of images from different modalities. Registration techniques include image intensity-based, feature-based (fiducial landmarks or anatomical landmarks), surface-based, and stereotactic frame-based techniques[4]. For example, x-ray fluoroscopy, which accounts for more than 90% of intraoperative imaging for guidance, generates 2D projection images. An interventional radiologist has to mentally register these 2D projection images to the 3D patient body to perform the procedure. This can create ambiguity and inaccuracies in the procedure. However, by registering a preoperative 3D magnetic resonance or CT image to the x-ray fluoroscopic image, 3D image information can be used to guide the procedure.

Another essential component of an image-guided system is the tracking device. A tracking system measures the real-time position of surgical tools and fiducials attached to the patient's anatomy. Visual virtual feedback can be provided by generating computer graphic images where these changing positions of the surgeon's tools are superimposed on the preoperative or intraoperative images. This helps the surgeon navigate inside the patient's body by using the overlaid images. The main tracking technologies used in image-guided procedures are optical and electromagnetic tracking. Optical tracking systems require an unobstructed line of sight between the tracked sensors and tracking device. Electromagnetic tracking system uses small sensor coils that can be embedded in instruments and does not have the line of sight limitation. Thus, electromagnetic tracking can be used to track instruments inside the body. However, electromagnetic tracking is affected by ferromagnetic objects in its working volume, and care must be taken to have a relatively metal-free environment. An example electromagnetic tracking system is shown in Figure 2.

To provide visual feedback to the clinician, computer graphic images are generated where registered information of the tracked instrument and preoperative or intraoperative image is presented in a meaningful manner. Visualization plays a key role in this regard. The visualization application typically displays the virtual anatomy around the location of the surgical instrument in real-time. Traditionally, slice-by-slice or multiplanar display views are used. The images can also be reformatted to create a view perpendicular to the direction or plane of the probe. With the increase in the resolution of medical images, surface and volume rendering techniques are commonly used to display 3D information of the surgical scene. Surface rendering techniques display surface structures identified within the image (the surface could be generated using contouring techniques). This may be augmented by texture mapping in which the original data is pasted on selective surfaces. For volume rendering, a ray-casting technique is commonly used where the displayed intensity of the image at each point is a function of the characteristics of the structures traversed by the ray. Advancement in these visualization algorithms and computing power has made interactive rendering of 3D scenes possible in surgical procedures.

## IGSTK SOFTWARE PROCESS

IGSTK is designed to develop safety-critical applications. This has led us to adopt a lightweight

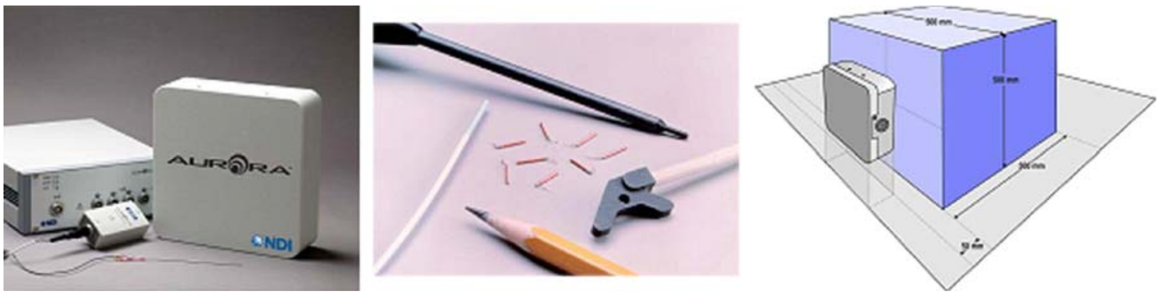


**Fig 2. Aurora electromagnetic tracking system components, sensors, and measurement volume. The left picture shows (from *left* to *right*) the control unit, sensor interface device, and electromagnetic field generator. The middle picture shows the sensor coils (*red objects* in the *middle* of the figure). The *right* picture shows the measurement volume which is a 500 mm cube starting 50 mm from the front of the field generator (Image courtesy of Northern Digital, Inc.).**

software process that emphasizes safety and robustness while also supporting geographically separated developers. The adopted software process is philosophically aligned with agile software development methods where lightweight, iterative, incremental, and test-driven development principles are applied. The IGSTK software process has four major components: source code version control system, build and release management, automated testing, and communication and documentation.

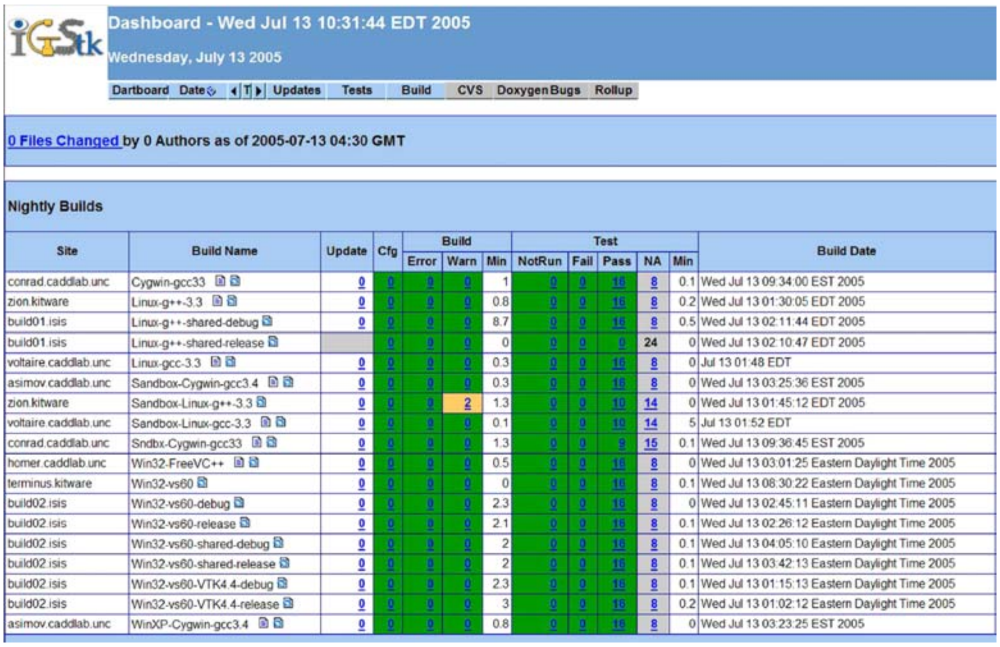
## Version Control System

Managing source code versions is essential in an iterative software development process. IGSTK uses the concurrent versioning system (CVS). CVS follows a client–server model where a server maintains a central repository for developers to check out, modify, and merge changes back to the code base. CVS allows simultaneous edits and branching and makes releases easier. IGSTK has two code repositories: a main CVS and a sandbox. The main CVS repository contains the version of the code that has been reviewed, well-tested, and approved for integration. The code in the main CVS repository adheres to all the quality software policies established in IGSTK. On the other hand, the sandbox is a testbed for developers to experiment on new ideas before integration with the rest of the toolkit, and it is subjected to less stringent quality requirements.

## Build and Release Management System

Configuring and managing the build process of a large project is a challenging task in software development. Large projects employ various tools and libraries. Developers use different compilers and development environments for coding. The build process should be able to coordinate all these tools. The task is particularly formidable for a cross-platform toolkit like IGSTK. The cross-platform project configuration task includes selecting an appropriate compiler, finding required packages and libraries, specifying include header file and library file paths, and selecting the appropriate compiler flags and options consistently across platforms.

The IGSTK configuration and build process is controlled by Cmake[5]. CMake is a cross-platform build system. CMake simplifies the configuration process by using platform-independent configuration files to generate native build files such as UNIX makefiles and Visual Studio workspaces for the user-selected compiler. CMake automates the configuration process and makes it possible to


Dashboard - Wed Jul 13 10:31:44 EDT 2005

Wednesday, July 13 2005

0 Files Changed by 0 Authors as of 2005-07-13 04:30 GMT

Nightly Builds

| Site | Build Name | Update | Cfg | Build | | | Test | | | | | Build Date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Error | Warn | Min | NotRun | Fail | Pass | NA | Min | |
| conrad.caddlab.unc | Cygwin-gcc33 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 16 | 8 | 0.1 | Wed Jul 13 09:34:00 EST 2005 |
| zion.kitware | Linux-g++-3.3 | 0 | 0 | 0 | 0 | 0.8 | 0 | 0 | 16 | 8 | 0.2 | Wed Jul 13 01:30:05 EDT 2005 |
| build01.isis | Linux-g++-shared-debug | 0 | 0 | 0 | 0 | 8.7 | 0 | 0 | 16 | 8 | 0.5 | Wed Jul 13 02:11:44 EDT 2005 |
| build01.isis | Linux-g++-shared-release | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 | Wed Jul 13 02:10:47 EDT 2005 |
| voltaire.caddlab.unc | Linux-gcc-3.3 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0 | 16 | 8 | 0 | Jul 13 01:48 EDT |
| asimov.caddlab.unc | Sandbox-Cygwin-gcc3.4 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0 | 16 | 8 | 0 | Wed Jul 13 03:25:36 EST 2005 |
| zion.kitware | Sandbox-Linux-g++-3.3 | 0 | 0 | 0 | 2 | 1.3 | 0 | 0 | 10 | 14 | 0 | Wed Jul 13 01:45:12 EDT 2005 |
| voltaire.caddlab.unc | Sandbox-Linux-gcc-3.3 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 10 | 14 | 5 | Jul 13 01:52 EDT |
| conrad.caddlab.unc | Sndbx-Cygwin-gcc33 | 0 | 0 | 0 | 0 | 1.3 | 0 | 0 | 9 | 15 | 0.1 | Wed Jul 13 09:36:45 EST 2005 |
| homer.caddlab.unc | Win32-FreeVC++ | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 16 | 8 | 0 | Wed Jul 13 03:01:25 Eastern Daylight Time 2005 |
| terminus.kitware | Win32-vs60 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 8 | 0.1 | Wed Jul 13 08:30:22 Eastern Daylight Time 2005 |
| build02.isis | Win32-vs60-debug | 0 | 0 | 0 | 0 | 2.3 | 0 | 0 | 16 | 8 | 0 | Wed Jul 13 02:45:11 Eastern Daylight Time 2005 |
| build02.isis | Win32-vs60-release | 0 | 0 | 0 | 0 | 2.1 | 0 | 0 | 16 | 8 | 0.1 | Wed Jul 13 02:26:12 Eastern Daylight Time 2005 |
| build02.isis | Win32-vs60-shared-debug | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 16 | 8 | 0.1 | Wed Jul 13 04:05:10 Eastern Daylight Time 2005 |
| build02.isis | Win32-vs60-shared-release | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 16 | 8 | 0.1 | Wed Jul 13 03:42:13 Eastern Daylight Time 2005 |
| build02.isis | Win32-vs60-VTK4.4-debug | 0 | 0 | 0 | 0 | 2.3 | 0 | 0 | 16 | 8 | 0.1 | Wed Jul 13 01:15:13 Eastern Daylight Time 2005 |
| build02.isis | Win32-vs60-VTK4.4-release | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 16 | 8 | 0.2 | Wed Jul 13 01:02:12 Eastern Daylight Time 2005 |
| asimov.caddlab.unc | WinXP-Cygwin-gcc3.4 | 0 | 0 | 0 | 0 | 0.8 | 0 | 0 | 16 | 8 | 0 | Wed Jul 13 03:23:25 EST 2005 |

**Fig 3. Nightly dashboard for multi-platform quality control.**

develop a cross-platform toolkit without too much additional effort.

IGSTK has official and interim release cycles. The official releases are generally planned once a year. Official releases contain completed major functionalities of the toolkit such as new components. In between official releases, the team aims for interim releases approximately 2 or 3 months. Interim releases typically include new features for some components. Release information is posted on the IGSTK Wiki and the website ( http://www.igstk.org ).

## Automated Testing

Incremental development relies on a continuous and stable software code base. Testing is critical to maintain the quality of the software. Incremental development can benefit greatly from continuous and automated testing. Automated testing provides instant feedback on the impact of new additions to the code base or bug fixes on the rest of the toolkit. Because IGSTK is a cross-platform toolkit, it is tested on various combinations of hardware, operating systems, and compilers, and the results are reported back to the developers. This is made possible by using the combination of CTest and the Dart system. CTest is one of the components of CMake, and it is intended to orchestrate the configuration, building, and testing of the software by gathering and submitting the results of these processes to a Dart server.

Dart is an open source infrastructure tool designed for software quality control in the context of a geographically distributed development team. In the Dart system, the software that is being tested is run on multiple combinations of hardware, operating systems, and compilers, and the results are submitted to a central server, where they are made available for review and feedback in a publicly accessible web page. In addition to compilation and build results, code coverage, unit test, and dynamic analysis results are posted on the Dart dashboard. This mechanism ensures a continued quality control of the software during the development stage. The dashboard accepts nightly, experimental, and continuous build results. Every night at a specified time, the entire IGSTK toolkit is built on multiple machines, and all the tests are run, and the results are posted on the dashboard.

Figure 3 shows a screenshot of a nightly IGSTK dashboard. During the day, if new changes are committed to the repository, continuous builds are submitted automatically to the dashboard. In this way, changes are immediately tested without waiting for the nightly build, and the developer can easily trace problems that the new changes may have caused in the rest of the toolkit. Experimental builds are used to test locally modified versions of the toolkit. This provides the developer with feedback about the impact of the local change on the rest of the toolkit before committing it to the repository.

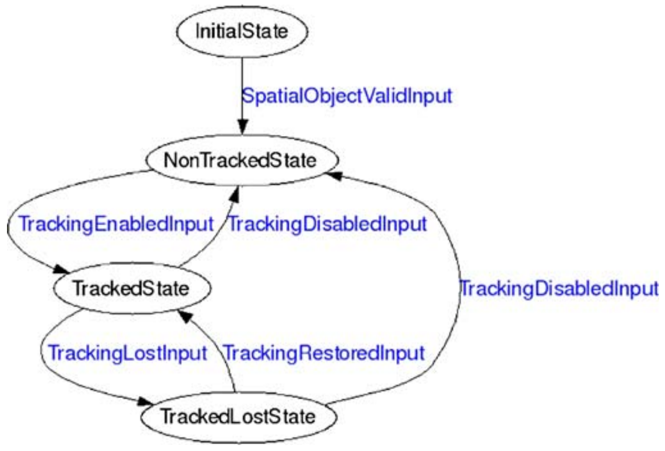


**Fig 4. State machine for IGSTK tracker component showing the four states in black and the transitions in blue.**

### Communication and Documentation

Communication is essential for the success of a collaborative project, particularly for a geographically distributed development team. The IGSTK software process was designed to facilitate effective communication among developers and project leads. Biweekly teleconferences, idea sharing using Wiki pages, and mailing lists are powerful and effective tools for communication. Technical topics are discussed in great detail regularly during the teleconferences. Furthermore, the dashboard is reviewed at each teleconference to maintain the quality of the software.

Documentation is essential for the efficient and continued use of a toolkit. IGSTK follows the literate programming philosophy. Literate programming encourages developers to include human-readable documentation in the source code. IGSTK uses the Doxygen tool for automated document generation from the source code. Doxygen employs a simple mark up language that is embedded as comment statements in the source code. This mechanism has two main benefits. First, contrary to traditional software development approaches, developers will not have to spend time after code development writing documentation. Instead developers type the documentation as they continue implementing the code while the ideas are still fresh and while they are in the mindset of the code logic. In this way, the documentation will be ready when code development is finished. Secondly, when developers modify the code to add a new feature or fix a big, they are required to update the markup comments accordingly so that the documentation will stay current with the code. In IGSTK, the document generation process is also integrated with the nightly build system. Every night, the documentation is generated and made accessible from the dashboard. Hence, users always have access to the latest documentation.

To summarize the software process, here are the ten best practices adopted by the IGSTK team[6]:

1) Recognize that people are the most important mechanism available for ensuring high quality software
2) Facilitate constant communication among developers
3) Produce iterative releases
4) Manage source code from a quality perspective
5) Focus on 100% code and path coverage at the component level
6) Emphasize continuous builds and testing
7) Support the development process with robust tools
8) Manage requirements iteratively in lockstep with code management
9) Focus on meeting exactly the current set of requirements, and
10) Evolve the development process.

## IGSTK ARCHITECTURE

IGSTK supports development of image-guided surgery applications that are classical examples of safety-critical applications. As a mechanism for preventing patient's harm, the architecture is based on the concept of safety-by-design. In particular, the IGSTK team followed a component-based

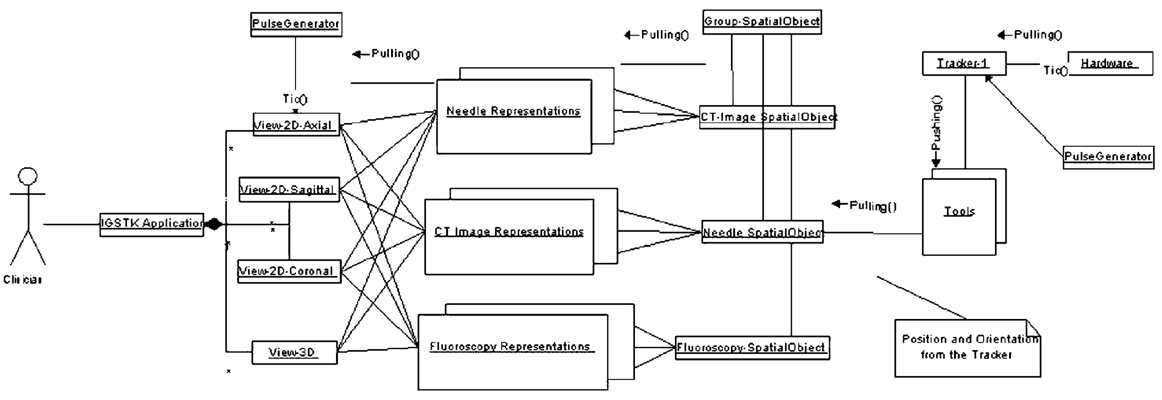


**Fig 5. IGSTK component architecture.**

architecture[7] and used state machine[8] software design methodologies. Every IGSTK component has a well-defined set of features that are governed by state machines. The state machine ensures that the component is always in a valid state and all state transitions are valid and meaningful. Figure 4 shows the state machine implementation for the tracker component (the IGSTK components will be described later in this section).

In a component-based architecture, the main capabilities are broken down into functional components with well-defined interfaces for inter-component communication. This type of architecture has several benefits when developing a reliable and robust toolkit. Breaking down the complex functionalities into smaller pieces makes implementation manageable and makes it possible to enforce higher standards of quality control. It is much easier to thoroughly test small components with well-defined interfaces than to test medium or large size components with myriads of features and convoluted interfaces. Consequently, complexity and implementation details are encapsulated in the component level and hidden from the application developer or user. This encapsulation allows IGSTK to better manage the inherent complexity of an image-guided surgery system. Furthermore, testing is easier at the component level compared to traditional procedural software thereby increasing the quality of the software. Application developers can also use some of the components independently in their application without the need to integrate the full toolkit. This should increase the adoption of the software and subsequently benefit the user community. Lastly, a component-based architecture is suitable for structured toolkit extension. This will encourage users to extend the toolkit by adding new components and by contributing them back to the community.

A state machine is defined by a set of states, a set of inputs, and a set of directed transitions from state to state. Each IGSTK component is subjected to state machine control. Incorporation of state machines has enhanced the reliability of the toolkit for the following reasons. State machines can ensure that the components have deterministic behavior at all times and are always in a known and error-free state. Formal validation[9] of the software can be undertaken because inputs, states, and transitions are well defined and are finite. This framework is suitable for automated testing. Interaction between the user and other applications can be explicitly defined using state machines reducing the possibility of design and implementation flows in application development. In summary, state machines ensure safety and reliability, cleaner design, application programming interface (API) simplicity, consistent integration pattern, and allow quality control. State machines can also prevent the misuse of components and help manage complexity, traceability, and testing.

IGSTK components are implemented using the C++ programming language. The component implementations are mostly based on ITK and

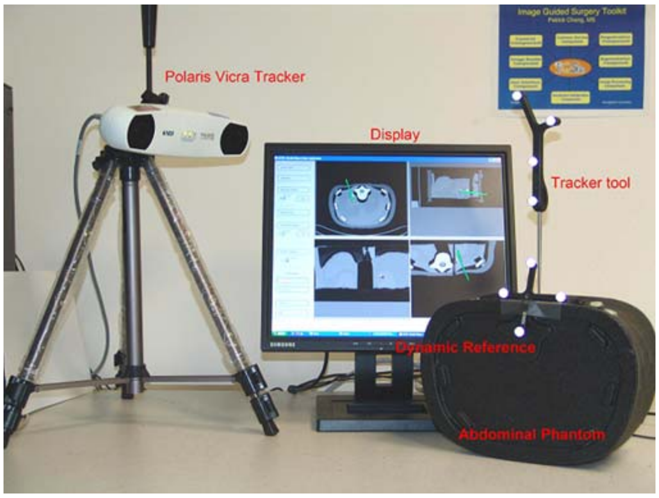


**Fig 6. Needle biopsy application system setup.**

VTK classes subjected to a strict state machine control. IGSTK components however do not expose ITK or VTK classes in their public interface. This exposure is avoided to increase the level of safety of the components. Interactions with the ITK and VTK classes are done only inside the IGSTK components and only under the supervision of the state machine.

The IGSTK architecture is shown in Figure 5. The main components in IGSTK are presented in the following list, and a short description of each component is given:

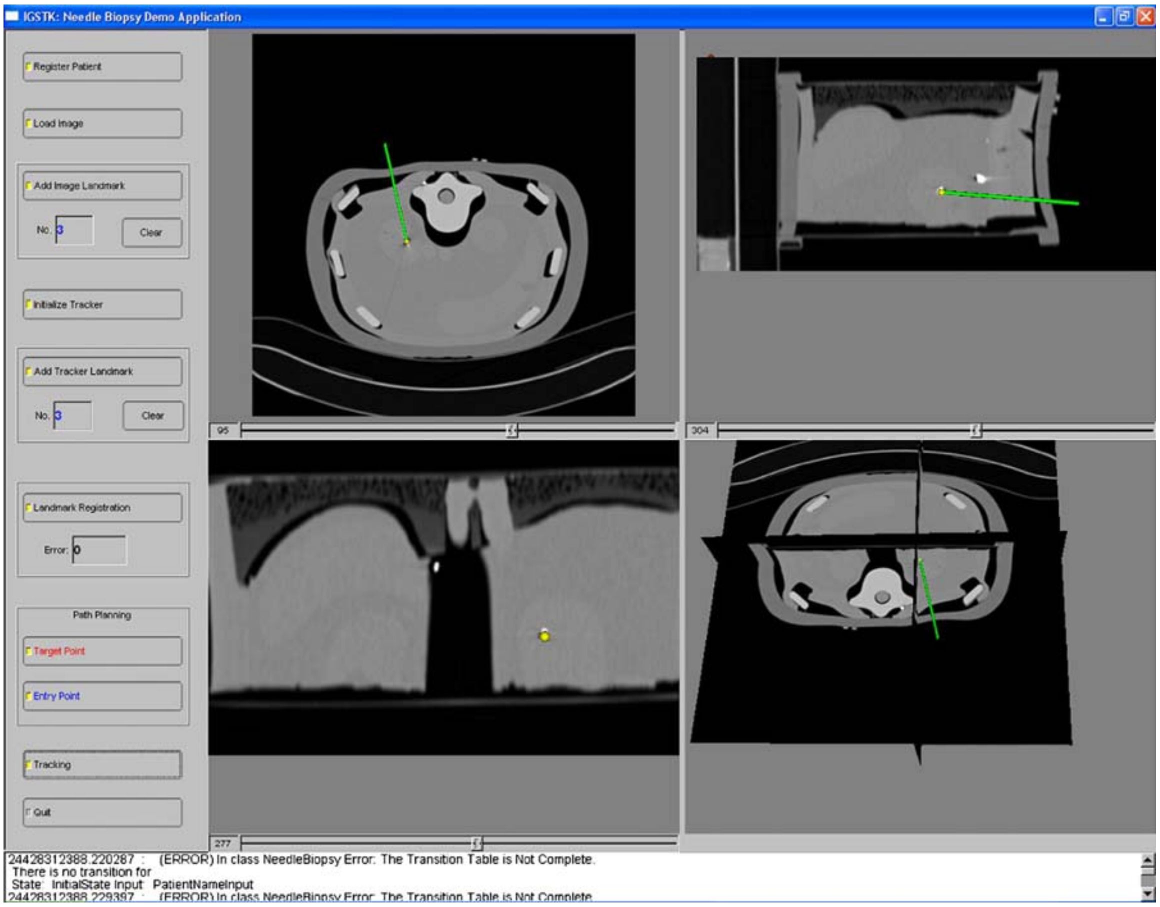
- View (Display)
- Spatial objects (geometric representation)
- Spatial object representation (visual representation)
- Trackers
- Readers

## View (Display)

Viewers display the graphical representations of the renderings of surgical scenes. Surgeons perform their task by visualizing the information provided in the viewers. Furthermore, the view components serve as a link between the graphical user interface library and the rest of the IGSTK toolkit.

## Spatial Objects (Geometrical Representation)

Spatial objects define a common structure for geometrical objects in IGSTK. The spatial objects hold the shapes and physical locations of objects in the surgical environment. IGSTK spatial objects encapsulate ITK spatial objects in a restrictive API. Spatial objects provided by IGSTK include objects such as axes, boxes, cones, cylinders, images, ellipsoids, meshes, tubes, and vascular networks.



```
24428312388.220287  :    (ERROR) In class NeedleBiopsy Error: The Transition Table is Not Complete.
 There is no transition for
 State:  InitialState Input:  PatientNameInput
24428312388.229397  :    (ERROR) In class NeedleBiopsy Error: The Transition Table is Not Complete.
```

**Fig 7. Graphical user interface for the needle biopsy application.**

## Spatial Object Representation (Visual Representation)

Spatial object representations characterize the graphical representations of the spatial object. The graphical representations dictate how an object should be displayed on the screen. This will include specifying color, opacity, and other rendering properties.

## Trackers

The tracker component handles the communication between tracking tools and tracking devices to get position, orientation, and other relevant information from surgical instruments present in the scene. The tracking component encapsulates the tracking tool, tracking device, and all static and dynamic information associated with tracking.

## Readers

Readers bring data into the scene generation and representation process. The most important readers in IGSTK are the Digital Imaging and Communications in Medicine (DICOM) image reader classes. These classes are used to read preoperative and intraoperative scans for surgical planning and guidance. Validity check logic is implemented in these classes to avoid incorrect file reads and 3D volume generation. Additional readers are available also for loading mesh and calibration data.

In addition to the above main components, IGSTK has a collection of infrastructure and service classes. The infrastructure classes include state machines, events, pulse generators, and real-time clock generator classes. Service classes include loggers, registration, and calibration classes. Loggers are useful for post-analysis of surgical procedures and recovery from a failure. Integrating loggers into applications streamlines and expedites the application development process and also provides a suitable framework for verification and validation.

Another service class in IGSTK is the registration class. This class computes the spatial transformation between the patient and the coordinate systems of the multiple image data sets that may be present in the scene, including preoperative and intraoperative images. Finally, a calibration class is available to measure the most important point of a surgical instrument relative to the position of the tracked element on the tool attached to the instrument.

## Putting It All Together

To demonstrate how the components can be integrated to develop an application, we will discuss an example image-guided needle biopsy application developed using IGSTK.

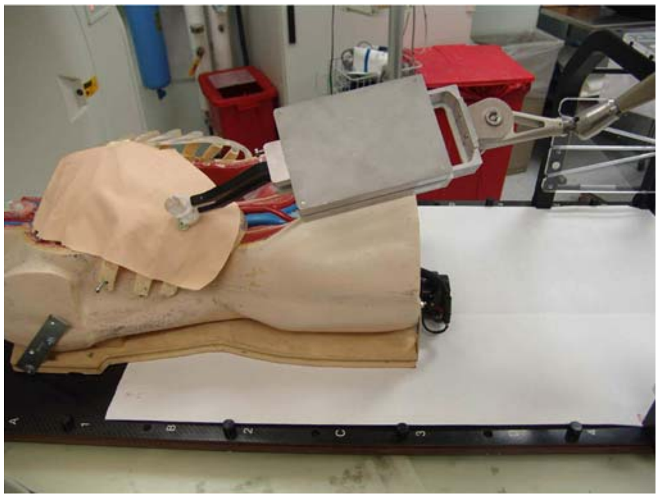


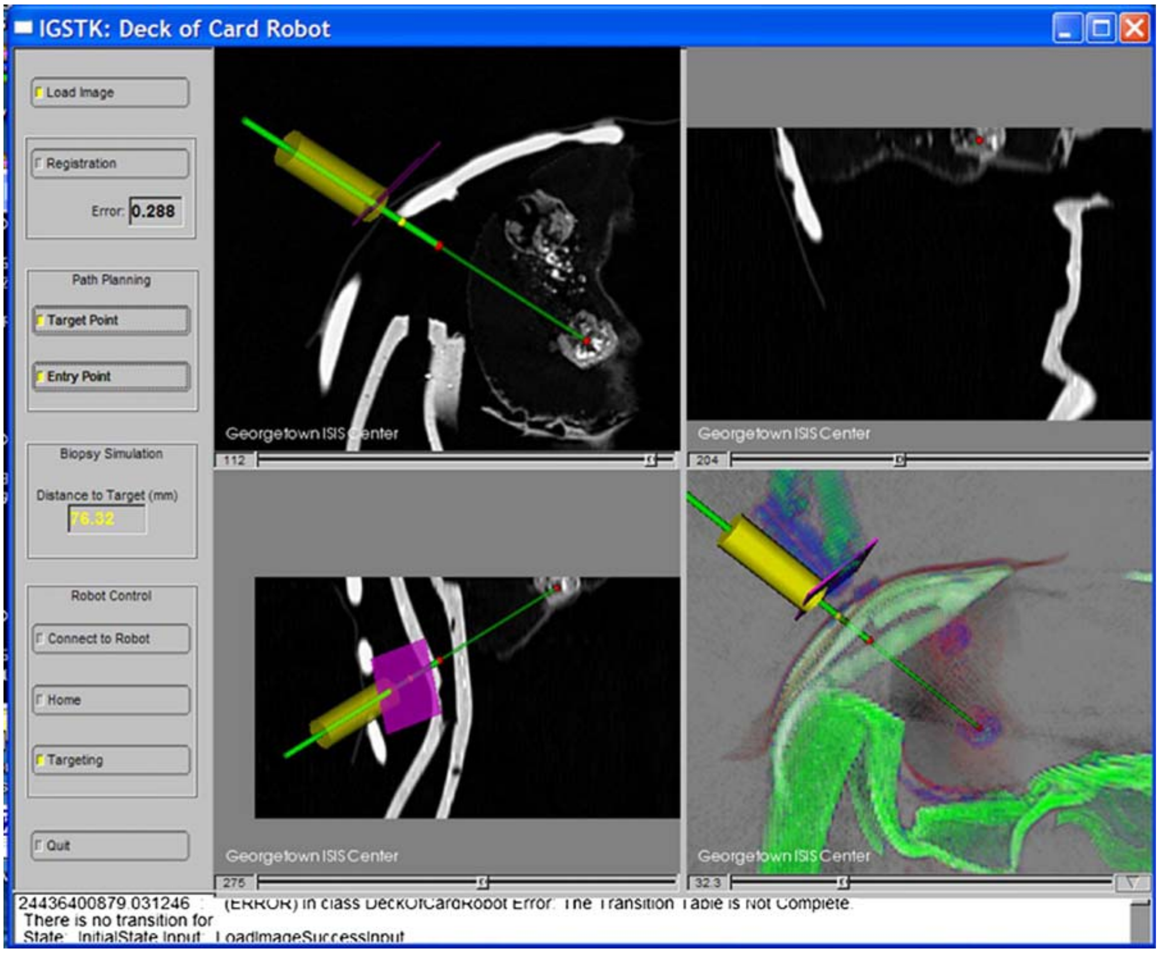
**Fig 8. Robot assisted needle placement phantom study.**

Needle biopsy is a medical procedure intended >to take a sample of tissue from a lump or tumor or other abnormal growth in the body with the purpose of performing a pathological analysis. An interventional radiologist will insert a needle into the tumor and take a tissue sample. This is a common procedure for cancer diagnosis. It is critical to ensure that the needle reaches the target that has been identified from the images.

In this application, the patient is positioned on the CT table, a CT image is acquired, and the location of the pathological tissue is identified on the images. A landmark-based registration technique is then used for registering the image coordinate system to the patient coordinate system. During the biopsy procedure, a computer graphics-generated representation of the surgical scene is presented in the display and is updated with the current position and orientation of the needle as they are continuously reported by a tracker. The workflow of this application is outlined below.

1) Record patient demographic information
2) Acquire and transfer CT image to the image-guided system
3) Identify landmark points in the image using the mouse. A minimum of three noncollinear landmark points are required.
4) Initialize the tracking device
5) Identify the corresponding landmarks in the physical body using the tracker pointing device
6) Perform registration to compute the transformation from patient to preoperative image
7) Identify entry point and target position for needle path planning
8) Start tracking.

When tracking is started, an updated position of the needle is displayed and overlaid on the CT



```
24436400879.031246 : (ERROR) in class DeckOfCardRobot Error: The Transition Table is Not Complete.
 There is no transition for
 State: InitialState Input: LoadImageSuccessInput
```

Fig 9. Graphical user interface for the robot assisted needle placement application.

image. The radiologist can then manipulate the needle, watching the virtual image, until the needle reaches the target. A confirming CT image can then be acquired.

For this example application, a Polaris Vicra optical tracker (Northern Digital, Ontario, Canada) was used. Testing was performed using an abdominal phantom (CIRS Model 57, Norfolk, VA). The application set up is shown in Figure 6. Figure 7 shows the user interface, which consists of a control panel and four standardized views: axial, sagittal, coronal, and a 3D view. The four view windows show CT images of the abdominal phantom with the overlay of the biopsy needle path. The green cylinder represents the needle as tracked by the Polaris Vicra Tracker. The four views automatically re-slice and update the images to show the needle tip position as it moves in the patient's anatomy.

## BUILDING AN IGSTK COMMUNITY

The continued success and viability of an open source toolkit depends on strong user community support. High quality design and architectural robustness are significant factors that drive the adoption of open source software. However, for long-term success and continuous evolution of the software, contributions from the user community are equally vital. Beyond the first few years of development supported by funding agencies, the lifetime of the toolkit depends on dedicated users and volunteers who care about the toolkit. Hence, as part of the toolkit development effort, toolkit creators should treat building and supporting the user community as a very important task. Without a committed user community, any open source software fades away after just few years of existence. With this understanding, the IGSTK team has taken several key steps to build a strong IGSTK community.

Complete documentation encourages users to evaluate the toolkit and contribute modifications and bug fixes. Automated documentation generation techniques simplify this task in IGSTK. Users have access to the latest API documentation from the IGSTK website and from the dashboard. Furthermore, to provide a detailed description of the toolkit and technical explanation of the components, the IGSTK team has written and published a book.[10] The book is available in a PDF format for free download at the IGSTK website ( http://www.igstk.org ).

A users' mailing list has also been created. Questions posted in this mailing list are promptly answered by the developers of the toolkit. Furthermore, users post bug reports, make feature requests, and contribute suggestions on how to improve the toolkit. These suggestions are seriously studied by the development team, documented in the bug tracker, and considered for implementation as time and resources permit.

Demonstrations of the toolkit is another key dissemination effort undertaken by the group. The needle biopsy application described above was demonstrated at the SPIE Medical Imaging Conference in 2006 and 2007 at San Diego, CA. Similarly, at the 2007 Society of Medical Innovation and Technology Conference, a robotically assisted needle placement application was demonstrated. The application guides the insertion of a needle using the robot. Figures 8 and 9 show the robot set up and a screenshot of the graphical user interface. The application demonstrations have helped to introduce the toolkit to the medical research community. Related to this effort, tutorial sessions on IGSTK have been offered at the SPIE Medical Imaging conference as part of the medical image analysis course using open source software.

What users are allowed to do with any open source software is defined by the copyright holder in the specific terms of the distribution license. For this reason, the selection of an appropriate license for the toolkit is also essential in building a strong user community. The copyright of IGSTK is held by the Insight Software Consortium ( http://www.insightsoftwareconsortium.org ). IGSTK is released under a Berkeley Software Distribution-like license, which allows the use of the software free of charge for academic and commercial applications. It also allows users to redistribute the software, modify it, and distribute the modifications without requiring permissions from the copyright holders. An example of the IGSTK commitment for building a strong community is the new collaboration that was recently established with the SINTEF Health Research Center at Norway. The center conducts cutting edge research on developing advanced navigation and visualization technologies for image-guided surgery.[11] The group is now actively involved in the development of new components of the IGSTK toolkit.

## CONCLUSION

Image-guided interventions are increasingly becoming the medical procedure of choice among patients and clinicians. They cause less trauma to the body and patients recover more rapidly from these procedures. Software is a critical component of such systems. Developing reliable software for such safety-critical applications is a challenging task. Oftentimes, research institutions invest large amounts of resources to build basic software infrastructures to develop these applications. With the availability of the IGSTK toolkit that contains all the components needed to build image-guided applications, researchers will be able to focus their resource on the main scientific problems. IGSTK is designed based on software principles intended to ensure reliability and patient safety in medical applications. The automated build and test management system deployed in IGSTK makes it easy to accept and integrate contributions from the user community while maintaining the high software quality standards established in the project. This is essential for the continued success and viability of open source software.

## ACKNOWLEDGMENTS

This project is a collaboration between Georgetown University, Kitware, Arizona State University, and Atamai. All of the software is freely available for download and can be used in research or commercial applications. More information can be found on the website at http://www.igstk.org.

This work was funded by NIBIB/NIH grant R01 EB007195 under project officer John Haller. Additional support was provided by US Army grant W81XWH-04-1-007, administered by the Telemedicine and Advanced Technology Research Center (TATRC), Fort Detrick, MD. The content of this manuscript does not necessarily reflect the position or policy of the US Government. We thank our other collaborators throughout the project, including Rick Avila and Will Schroeder of Kitware; Roland Stenzel of Georgetown University; Geir Arne Tangen, Ole Vegard Solberg, Arild Wollf, Torleif Sandnes of SINTEF Health Research, Medical Technology (and the National Centre for 3D Ultrasound in Surgery), Trondheim, Norway; Ivo Wolf of the German Cancer Research Center, Heidelberg; Peter Kazanzides and Anton Deguet of Johns Hopkins University; Nobuhiko Hata of Brigham and Women's Hospital; Sohan Ranjan of GE Research; Mihai Mocanu of the University of Craiova; and Matt McAuliffe and Terry Yoo of the NIH.

## REFERENCES

1. Schroeder W, Martin K, Lorensen B: The Visualization Toolkit: An object-oriented approach to computer graphics, 4th edition. Clifton Park, NY: Kitware Inc., 2006

2. Ibanez L, Schroeder W: The ITK Software Guide, 2nd edition. Clifton Park, NY: Kitware Inc., 2005

3. Forrester Consulting: Open source software's expanding role in the enterprise, A Forrester Consulting study commissioned by Unisys Corporation, Forrester Research Inc., 2007

4. Hajnal JV, Hill D, Hawkes DJ: Medical Image Registration. Boca Raton, FL: CRC Press LLC, 2001

5. Martin K, Hoffman B: Mastering CMake: A Cross-Platform Build System, 3rd ed. Clifton Park, NY: Kitware Inc., 2006

6. Gary K, Blake MB, Ibanez L, Gobbi D, Aylward S, Cleary K: IGSTK: An open source software platform for image-guided surgery. IEEE Computer 39(4):46–53, 2006

7. Blake MB, Cleary K, Kim HS, Ranjan S, Gary K, Jomier J, Aylward S, Ibanez L: Component-Based Design and Development for Robust Medical Applications, High Confidence Medical Device Software and Systems (HCMDSS) Workshop, 2005

8. Ibanez L, Jomier J, Gobbi D, Avila R, Blake MB, Kim H-S, Gary K, Aylward S, Cleary K, IGSTK: A State machine architecture for an open source software toolkit for image-guided surgery applications, Insight Journal—MICCAI Open-Source Workshop, 2005

9. Gary K, Kokoori S, David B, Otoom M, Blake MB, Cleary K: An Architecture Validation Toolset for Ensuring Patient Safety in an Open Source Software Toolkit for Image-Guided Surgery Applications, Insight Journal—MICCAI Open-Source Workshop, 2006

10. Cleary K, IGSTK Team: IGSTK: An Open Source C++ Software Library. Gaithersburg, MD: Signature Book Printing, 2007

11. Rasmussen IA, Lindseth F, Rygh OM, Berntsen EM, Selbekk T, Xu J, Nagelhus Hernes TA, Harg E, Haberg A, Unsgaard G: Functional neuronavigation combined with intra-operative 3D ultrasound: Initial experiences during surgical resections close to eloquent brain areas and future directions in automatic brain shift compensation of preoperative data. Acta Neurochir(Wien), 149(4):365–378, 2007