3D SLICER

Steve Pieper[1,2], *Michael Halle*[1], *Ron Kikinis*[1]
[1]Surgical Planning Laboratory, Brigham and Women's Hospital; [2]Isomics, Inc.

ABSTRACT

To be applied to practical clinical research problems, medical image computing software requires infrastructure including routines to read and write various file formats, manipulate 2D and 3D coordinate systems, and present a consistent user interface paradigm and visualization metaphor. At the same time, research software needs to be flexible to facilitate implementation of new ideas. 3D Slicer is a project that aims to provide a platform for a variety of applications through a community-development model. The resulting system has been used for research in both basic biomedical and clinically applied settings. 3D Slicer is built on a set of powerful and widely used software components (Tcl/Tk, VTK, ITK) to which is added an application layer that makes the system usable by non-programmer end-users. Using this approach, advanced applications including image guided surgery, robotics, brain mapping, and virtual colonoscopy have been implemented as 3D Slicer modules. In this paper we discuss some of the goals of the 3D Slicer project and how the architecture helps support those goals. We also point out some of the practical issues which arise from this approach.

1. INTRODUCTION

Medical image computing applications are complex pieces of software requiring a common set of base functionality as well as the ability to be customized for specific clinical applications. In a research environment, it is often necessary to create prototype environments that allow exploration and refinement of a new algorithm or concept in the context of a complete functional end-user application. The 3D Slicer project [1] (or simply 'Slicer') began as way to provide a common research platform with basic functionality and has evolved to support a wide variety of clinical applications. We approached this problem first from the perspective our own laboratory's requirements, but quickly realized that building a larger community of users and developers had the potential to create a more powerful and useful software environment. Following the philosophical model of Open Source software, we have created an infrastructure to manage the project and to encourage community involvement. The resulting software environment has been used as the basis for a number of scientific research efforts which provide the funding for Slicer's ongoing software engineering. There have been over 4000 registered downloads of Slicer not including developer access. The Slicer user's email list contains 166 subscribers; the developer's list 117 subscribers. There are about a dozen active developers with write access to the source code repository and a comparable number of module developers.

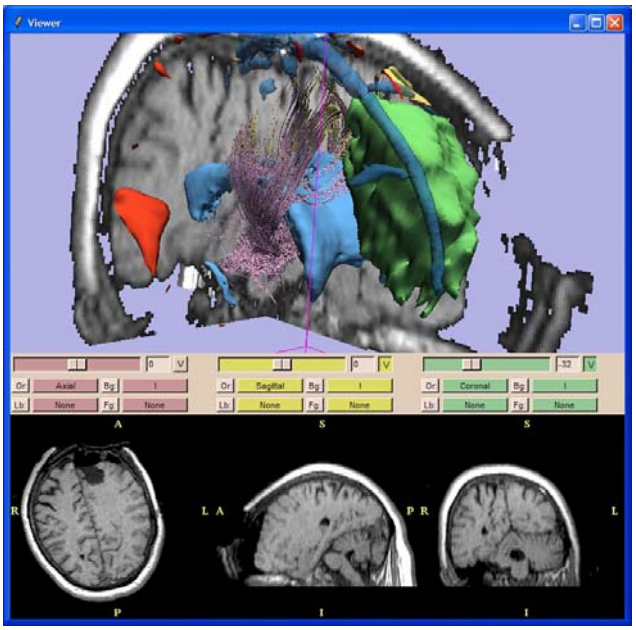


Figure 1 Sample 3D Slicer display showing intra-operative MRI slices and 3D reconstructions. (Data courtesy Dr. Ion-Florin Talos, Brigham and Women's Hospital).

2. SLICER GOALS AND NON-GOALS

To create a system meeting those requirements, Slicer was designed and continues to evolve with several goals and "non-goals" in mind. We use the term "non-goals" to refer to considerations that may be driving factors for other software development efforts but have been explicitly excluded from Slicer's objectives.

Slicer goals:

- to establish a common development platform for researchers within a clinical research environment;

- to provide users with a familiar user interface to perform image processing and visualization tasks;
- to establish a set of conventions for data handling and exchange that both developers and users can adopt when there is no overriding reason not to do so;
- to encourage transfer of algorithm and visualization techniques from developers to users for evaluation, refinement, and use;
- to foster information exchange and collaboration between different researchers, departments, and institutions, locally and world-wide; and
- to minimize the different costs of entry and membership in the developer and user community.

Slicer “non-goals”:

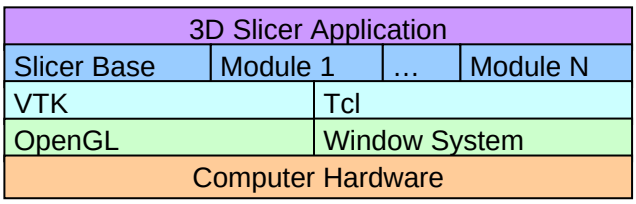
- not a goal to create a self-supporting revenue stream based on software sales or support;
- not a goal to further sales of a required commercial software package or hardware device;
- not a goal to lock users or developers into a single software platform;
- not a goal to protect intellectual property by limiting access to software code or internal functionality;
- not a goal to contractually guarantee clinical accuracy or reliability for research code;
- not a goal to have the code FDA-approved; and
- not a goal to provide all software components written internally "from scratch" in their entirety.

Slicer's non-goals may well be valid, even necessary goals for a commercial software company. It is also possible that a commercial third-party might use Slicer or parts of the Slicer code to help meet goals different from ours. By consciously freeing ourselves from these commercially-oriented goals, we have furthered Slicer's fundamental goals by encouraging a world-wide community of talented developers and skilled users to both use and contribute to the open platform. Also, while we cannot guarantee anything about the accuracy or reliability of the code, we believe that the software engineering infrastructure provides a high level of quality that is sufficient for advanced biomedical research.

Some of our greatest challenges, from both a software and social engineering perspective, have been to create mechanisms that take full advantage of the creativity and expertise of our contributors while maintaining Slicer's stability. Our developers are generally experts in their own fields, not in Slicer development. They are literally all over the world, using a variety of operating systems and computer hardware. They work for corporations, universities, hospitals, or even just themselves. Without some guidance, control, and limits, the entire process risks devolving into chaos.

We have found that several elements are essential to managing this distributed developer and user environment. The first is good communication. Without it, no standard of work is possible, efforts become duplicated, and individuals feel isolated rather than part of a larger team. We make heavy use of both developer and user electronic mail lists to discuss current and future issues with Slicer. For those developers who are local, a semi-weekly meeting provides an opportunity to hold face-to-face discussions. Notes from these meetings are available through the mailing list and on Slicer's web page. The web page also provides a central focus for all documentation for both users and developers. Access to the software repository is available to developers both through the web and through CVS (Concurrent Version System [2]), permitting authorized software access and updates from anywhere on the network. These central services (mailing lists, web page, software repository) are maintained by the same professional staff that is responsible for core Slicer development; these computational and personnel resources form the stable structure upon which Slicer development is built.

The second most important principle of our work is modularity. Modular systems allow developers working in relative isolation to produce elements of value that other people can use. Modularity is also essential for software stability in quickly-developing systems: a small change by any one developer should have minimal or no adverse impact on other developers or users. Modularity encourages the concept of a toolbox that can be tailored to particular tasks. Finally, modular systems allow developers to concentrate on their own areas of expertise, enabling them to understand, implement and test their own software elements without extensive knowledge of the larger platform.



Modularity in Slicer is done using layers of abstractions and componentized functional units. Slicer is layered to a large extent on VTK (Visualization Toolkit [3]), a freely-available visualization toolkit written in C++ from Kitware, Inc. [4]. Slicer extends VTK's base functionality with C++ modules designed for medical imaging. VTK also provides bindings to most of its functionality for interpreted languages such as Tcl [5]. Slicer uses these bindings to provide a common library of Tcl script code and to implement Slicer's cross-platform user interface written in the Tcl/Tk windowing toolkit. These foundation

layers also provide the parsing semantics for MRML, Slicer's XML-based data exchange file format. Slicer's component architecture is based on the concept of modules that can provide new user interface components, new software services written in Tcl or C++, or a combination thereof. Modules can be developed individually and do not require access to all other modules to compile or function. Modules are loaded into Slicer at runtime using Tcl's package mechanism and the dynamic loading capability of modern operating systems.

A modular software structure permits variations in development styles, timetables, or licensing structures between different modules. Policies for one module do not in general dictate the policies for any other. It is possible, for instance, that the compiled VTK implementation class for a 3D Slicer module could be distributed in binary form only, without source code, and used to provide new functionality. While such a policy might seem to be contrary to the spirit of an open platform, the choice is a social one; 3D Slicer's software architecture does not preclude it. Such flexibility can actually be quite useful in an academic environment: users can gain the benefits of modules using newly developed algorithms, while module writers can temporarily protect their work prior to publication.

3. ISSUES AND LIMITATIONS

While the project design and implementation have been broadly successful at meeting Slicer's goals, a number of specific issues require ongoing attention.

3.1. Non-Research Clinical Applications

Although Slicer is not intended to be an FDA regulated medical device, validation of the design and verification of the implementation are important aspects of any software package, particularly one to be used as the basis for scientific studies to be published in the literature and for clinically-oriented research. This is a particular challenge in the face of the diverse development community process described above. Slicer does rely on the automated testing infrastructure provided by the packages on which it relies: Tcl/Tk, VTK, and ITK (Insight Toolkit [6]) all include regression tests as a major part of the development process. One can point to anecdotal evidence from open source projects that the open development process identifies and fixes many bugs that might go undiscovered in proprietary packages. While we believe this trend to be true and that Slicer has benefited from that process, the evidence is not adequate to support claims of clinical accuracy.

To date, our only firm answer to the question of adequacy for clinical use is to forbid, in the software license agreement, any non-research use of the software and to state that: "In no event shall data or images generated through the use of 3D Slicer Software be used in the provision of patient care." See [1] for full text of software license agreement. These limitations are driven by the very real concern that addition of new features will have unintended consequences in other parts of the code, in spite of the developers' best intentions and the modular design of the software.

The limitation of use to research applications is a reflection of three aspects of the current state of the Slicer development process. First, there is no fixed set of functionality for the program as a whole to test against because features are being added and modified to suit the needs of the developers and users. This is particularly difficult due to the extensive and rich set of user interactions with the 3D environment; refining and improving these 3D interactions is a critical part of the research effort. Second, there is currently no individual or group whose job function is to perform exhaustive testing of the software to the level required for clinical use. Third, there is no entity with an interest in claiming or supporting a particular level of functionality or accuracy; that is, in typical FDA regulated medical device software a corporation makes certain claims that the software is suitable for clinical application and is held responsible for ensuring that the product lives up to those claims. Although some members of the development teams are part of hospitals that provide clinical care or corporations that develop medical devices, the institutions themselves specifically disclaim any responsibility for Slicer.

The three issues mentioned in the previous paragraph are not fundamental limitations and, as stated, are merely reflections of the current realities of the development community. One possible future development would be for a corporation to step forward and take on responsibility for refining and testing the software in support of particular claims of clinical functionality. Such an effort would no doubt require freezing certain aspects of the code and probably removing some functionality. While there may be specific intellectual property embodied in non-public Slicer modules, this approach would require extensive testing and bug fixing in the base code. Since the Slicer license agreement requires that changes to the base code be made publicly available, we believe this scenario would be beneficial to the Slicer development community by improving the robustness of the underlying software and by providing an avenue for possible clinical application.

3.2. Major Architectural Changes

Another consequence of the distributed nature of the development community is that over the years several

individuals and development teams have contributed major sections of the code and then moved on to other projects. Because some strong development guidelines were in place from the beginning of the project, most of the code is readable and maintainable. In particular, the C++ code in Slicer has benefited greatly from the implementation structure provided by VTK which provides a class hierarchy and build mechanism that has been adopted and extended within Slicer. Also, because C++ is used for the compute-intensive aspects of the program, while Tcl/Tk is used for the application logic and user interface, there is a natural division of the organizational structure common to the modules. As with any large program, certain aspects of the design become out of date as new requirements emerge. It is an ongoing challenge to incorporate new designs within the base code while maintaining compatibility with the modules; this reality has limited our ability to make major changes to the underlying architecture.

3.3. Engineering Effort and Funding

Another major consideration with any software development project is the availability of personnel, equipment, and other resources to ensure the ongoing viability of the project. With community-developed projects like Slicer, we benefit from the (often significant) contributions from the developers who provide bug fixes or new feature contributions as dictated by their own needs and interests. In spite of these valuable contributions, and in fact to make best use of them, some level of centralized administration and engineering effort is required. Our strategy in this regard has been to have Slicer adopted by various research projects and the standard visualization and image processing environment. These research efforts then serve as a focus for the development process and a spur to new functionality.

In particular, the Neuroimage Analysis Center (NAC) [7], a Biomedical Technology Resource Center funded by the National Center for Research Resources (NCRR) [8] at the National Institutes of Health is the organizational "home" of Slicer's development and administration. Through the NAC, Slicer is used actively in a variety of clinical research scenarios including neurosurgical planning, investigation of Alzheimer's Disease, multiple sclerosis, schizophrenia, and related conditions. The NAC is also collaborating with other NCRR-funded centers in the Biomedical Informatics Research Network (BIRN) [9] project, which provides additional funding for engineering and application development for the Slicer. Additional support comes from the National Science Foundation robotics project [10] and a newly forming Department of Defense project in support of combat casualty care. Each of these projects places new demands for added functionality but at the same time provides a context in which the base Slicer code can be improved and refined. The pace and open philosophy of Slicer development would simply not be possible without the support from these governmental funding sources.

4. CONCLUSION

The 3D Slicer project has proven to be an exciting and productive environment for advanced medical image computing projects. The goals articulated here have resulted in a platform where new functionality rapidly progresses from concept to implementation and where new multi-institution and multi-specialty collaborations are facilitated. This environment, appropriate for the research setting, is inherently less structured and restricted than conventional medical device software that used in clinical routine. While we openly acknowledge imperfections in Slicer's design and implementation, we believe that Slicer offers compelling abilities to improve and grow over time compared to commercially-licensed alternatives. We plan to continue working on new methods that will improve the quality and reliability of the code while preserving the support for creative contributions from the wider development community.

5. ACKNOWLEDGEMENTS

The authors would like to thank all the past and current Slicer developers. Special thanks to David Gering, Lauren O'Donnell and Nicole Aucoin for significant design and engineering contributions. This investigation was supported by NIH grant P41 RR13218 and the Biomedical Informatics Research Network (www.nbirn.net).

6. REFERENCES

[1] 3D Slicer home page http://www.slicer.org
[2] CVS home page http://www.cvshome.org
[3] VTK home page http://www.vtk.org
[4] Kitware home page http://www.kitware.com
[5] Tcl/Tk home page http://www.tcl.tk
[6] Insight Toolkit home page http://www.itk.org
[7] NAC home page
http://spl.harvard.edu/pages/projects/grants/nac
[8] NCRR Biomedical Technology Center home page
http://www.ncrr.nih.gov/biotech/btresctr.asp
[9] BIRN home page http://www.nbirn.net
[10] CISST home page http://cisstweb.cs.jhu.edu