SCIRUN/BIOPSE: INTEGRATED PROBLEM SOLVING ENVIRONMENT FOR BIOELECTRIC FIELD PROBLEMS AND VISUALIZATION

R.S. MacLeod D.M. Weinstein, J. Davison de St. Germain, D.H. Brooks[+]*, C.R. Johnson, and S.G. Parker*[*]

Scientific Computing and Imaging Institute University of Utah, Salt Lake City, Utah
[+] Department of Electrical and Computer Engineering, Northeastern University, Boston, MA

ABSTRACT

SCIRun is a general purpose problem solving environment that seeks to integrate the steps of preparing, executing, and visualizing simulations of physical and biological systems. The implementation of SCIRun is by means of an interactive dataflow network consisting of modules and data pipes exposed as a visual programming language. SCIRun also contains specific modules for bioelectric field simulations and visualizations and the combination of SCIRun with this package is known as BioPSE (www.sci.utah.edu/software/biopse). This software has been in the public domain since 2000 and in that time we have developed strategies for software development, engineering, testing, documentation, and training. We have also continued to expand the scope of the SCIRun/BioPSE package not only through our own codes but by constructing bridges to other systems, both open source and proprietary. We have also created a repository for relevant sample networks and datasets with the aim of allowing diverse groups to test and evaluate algorithms using identical data and to share their results with the community for comparison of performance and accuracy. We present here a summary of the software system and describe specific experiences and conclusions with regard to creating and managing a large open source software project carried out within a university setting.

1. INTRODUCTION AND DESIGN GOALS

SCIRun is a problem solving environment that seeks to integrate steps that are traditionally part of separate programs into a seamless software system [1, 2]. The underlying structure of SCIRun is a data pipeline that passes information through a series of processing elements, each of which is general in formulation but has a specific purpose within the application. A SCIRun program is therefore a collection of processing elements and the connections that link them. The usual interface to this structure is as a visual programming language composed of modules (the processing steps) and pipes (the connections) that join to form networks. The modules are the commands of the language and the pipes form the communication system passing information to and from those commands.

There are three pillars of the SCIRun vision: software re-use by leveraging algorithmic and visualization commonalities across disciplines, integration of all aspects of the simulation process in a "computational steering" environment, and multiple levels of access and re-use.

Fundamental to the vision that drove the creation of the SCIRun problem solving environment is that while each scientific discipline has its own terminology and its own specific problems of interest, from a broader perspective, their similarities often outnumber their differences. This is true, for example, when one considers the tasks of image processing, analysis and visualization. The clinical brain imaging community will have a significantly different terminology from that of the small animal imaging or cardiovascular communities, but the image processing, analysis, and visualization needs are attainable using similar fundamental algorithms, all of which come from a common (application independent) source in signal/image processing and computer science. Hence the goal of SCIRun was and is to identify and then implement in an integrated problem solving environment a set of common tools for scientific modeling, simulation, and visualization so that users can apply them to specific problems.

A second pillar of the SCIRun vision is integration of all aspects of scientific computation and visualization. The traditional (and still quite common) approach, for even moderately-sized problems, employs a suite of separate, often incompatible, programs. Typically, one creates a geometric model from image data with one set of programs. This model becomes input to a simulation program which, say, creates a linear system to solve a governing PDE. The next step is a linear system solver, followed by a separate program to visualize the results. At each inter-program interface there is logistical overhead. The biggest problem is that changing problem parameters usually requires another pass through most or all of this pipeline, thus creating long iteration loops. SCIRun integrates all these computing steps

[*] Support for this research has come from the NIH/NCRR for the Center for Bioelectric Field Modeling, Simulation, and Visualization, HL P41RR12553.

into a single connected process. Moreover, the user can intervene at any point in the process and see the consequences almost immediately. Achieving this flexibility requires a sophisticated, efficient infrastructure and multiple interactive access points.

A final component of the SCIRun vision is to provide multiple levels of access to the software. Each level will offer a different degree of control and thus flexibility but at an associated cost in user proficiency. The lowest level access is to the source code itself, which allows the developer to add almost limitless functionality but requires knowledge of C++ and the architecture of SCIRun. The next level is a visual programming interface to the network diagrams that define a SCIRun program. Networks link functional elements (modules) through data-pipes and the user can interactively add modules and connections as desired to create solutions. The highest level of access is in the form of dedicated applications that are built on top of the network diagrams. We have recently developed these "PowerApps" to provide access to the necessary control parameters of the underlying network through a customized visual interface so that the user is required only to understand features and options relevant to the particular application. At each of these levels of interaction, we provide documentation support and guidance so that the user can first decide on and then find the appropriate interface level.

2. SOFTWARE INFRASTRUCTURE AND IMPLEMENTATION

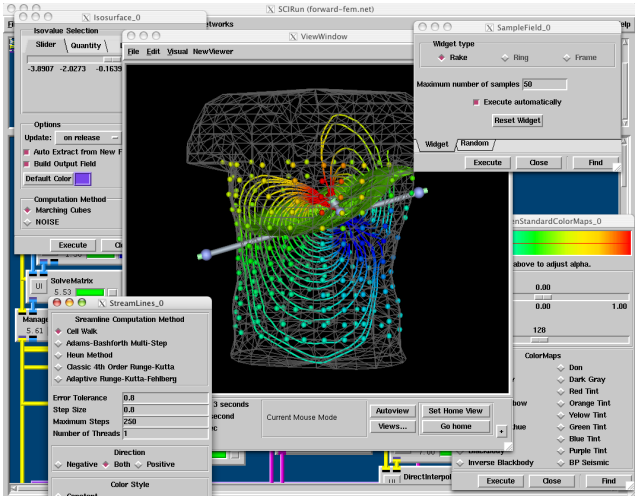

Fig. 1. BioPSE dataflow interface to the forward bioelectric field application. The underlying dataflow network implements the application with modular inter-connected components called modules. Data are passed between the modules as input and output parameters to the algorithms.

The SCIRun project began as a doctoral thesis of Dr. Steve Parker in the early 1990's [1] and has since grown and become an established part of a number of funded research projects and centers at the SCI Institute and other institutions across a broad range of application areas [2, 3]. Most visible of these is perhaps the NIH/NCRR funded Center for Bioelectric Field Modeling, Simulation, and Visualization (www.sci.utah.edu/ncrr), from which the BioPSE (for Biological Problem Solving Environment) tools have come. SCIRun/BioPSE thus consists of a complete system for solving bioelectric field problems such as computing the electrocardiogram (ECG) from known bioelectric sources in the heart or localizing sources of electrical activity in the brain from the electroencephalogram (EEG) or magnetoencephalogram (MEG) measured on or outside the head. In a completely different application, SCIRun also provides the infrastructure for simulating rapid fires and explosions in a large project funded by the Department of Energy's ASCII program called C-SAFE (www.csafe.utah.edu).

Historically, one of the major hurdles to SCIRun becoming a tool for the scientist as well as the engineer has been SCIRun's dataflow interface. While visual programming is natural for computer scientists and engineers who are accustomed to writing software and building algorithmic pipelines, it is overly cumbersome for application scientists. Even when a dataflow network implements a specific application, the user interface (UI) components of the network are presented to the user in separate applications windows, without any semantic context for the control of their settings. For example, Figure 1 contains an example of a dataflow network that solves a bioelectric forward problem using the finite element method. The figure shows a BioPSE network with the typical mix of visualization window, several UI's, and, in the background, the network diagram. Although SCIRun provides file browser UI's for reading in data, all of the file browsers in the dataflow network have the same generic presentation. As a result, there has not been a way in SCIRun to indicate that one file browser entry should identify an electrode input file and another should identify a finite element mesh file. Similarly, the context of the specific modules often makes the meaning of the user controls difficult to interpret. For example, in Figure 1 the text-entry field of the SampleField user interface in the upper right corner of the figure that is labeled "Maximum number of samples" is controlling the number of electric field streamlines that are produced for the visualization. In another context, the same user interface element could have a completely different meaning.

The most recent release 1.20 of BioPSE/SCIRun (in October, 2003) has addressed the complexity of the network interface through the introduction of "PowerApps". A PowerApp is a customized interface built atop a dataflow application network. The dataflow network controls the execution

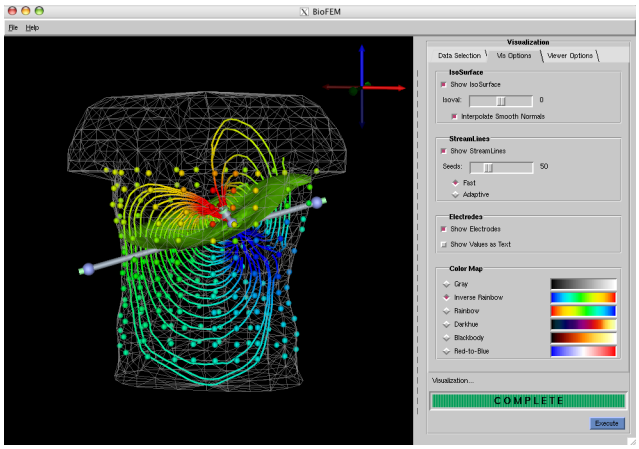

Fig. 2. The BioFEM custom interface. Though the application is functionality equivalent to the dataflow version shown in Figure 1, this PowerApp version provides a custom interface that is much simpler to navigate.

and synchronization of the modules that comprise the application. In a PowerApp, the associated network diagram and generic UI elements disappear (but are still available) in exchange for a comprehensive customized user interface. Each PowerApp has its own arrangement of user interface elements and visualization features and the organization of control parameters encourages a sequential approach that reflects the expected work flow of someone using the application.

One example of a PowerApp, released with the 1.20 version of SCIRun/BioPSE is called "BioFEM". BioFEM is built atop the finite element method forward solution network and provides a useful example for demonstrating the differences between the dataflow and PowerApp views of the same functionality. Figure 2 shows the user interface for BioFEM, running the same problem as in Figure 1. The visualization window is unobscured by UI elements and all user controls are consolidated in the control panel to the right. All controls include an explicit context through semantically linked labels and are grouped logically by means of the tabs along the top of the control panel. Application-specific textual tips (not shown) appear when the user places the cursor over any user interface element.

3. OPEN SOFTWARE FEATURES

SCIRun/BioPSE has been available as source code since 2000 and we have recorded several thousand downloads over a dozen code releases. In that time, we have gained experience in many aspects of open source software development, deployment, and maintenance. We summarize here a few aspects of those experiences and measures we have implemented from them.

3.1. Choice of platforms

The rapid growth of SCIRun/BioPSE has come in part because of the focused platform base we have supported to date. Unix offers a comprehensive set of development tools (gcc compilers, make, autoconf, CVS, *etc.*,) and, more importantly, a stable and relatively portable set of low level operating system features. The OpenGL graphics library, developed initially by Silicon Graphics (SGI) and now an open standard (www.opengl.org), offers the same consistency of interface to the graphics hardware. Thus the progression of SCIRun development has been from the Silicon Graphics Irix (Unix) to Linux to Apple OSX. Limiting SCIRun to these platforms has reduced the overhead required to maintain different versions and thus permitted more focus on increasing the capabilities and stability of the code. The associated disadvantage, of course, is the limitations to the user base of the software. The continuous pressure to expand the supported platforms to include Microsoft Windows is substantial and we are now in the planning phase of this transition and hence cannot yet robustly estimate the associated cost.

3.2. Open standards

One of the driving principals of the SCIRun design has always been to leverage open-standards as much as possible. Examples that have been part of SCIRun from the beginning include the use of pthreads for the implementation of multithreading (when available on the target platform); OpenGL for all rendering; and tcl/tk for all user-interface widgets. Other examples of standards that have become part of SCIRun as they appeared and gained dominance include the standard template library (STL), which required that SCIRun migrate away from the original native set of container classes, and XML, which is the markup language that describes user-interface port behavior for each module and the module level documentation. Two additional change, also in the planning or early implementation phases, is to replace the native SCIRun scene graph with an open standard such as OpenSceneGraph and to implement a Common Component Architecture (CCA), which provides defined interfaces between portable and reusable modules.

3.3. Bridging to other software system

A tenet of Unix and many open source software projects is to build on the work of others and thus it has always been a goal of SCIRun to make use of software outside our own code base. This desire has also come from the past frustration of trying to marry the best features of different software packages into an integrated whole. To achieve more seamless integration, we have developed a range of bridging mechanisms for linking SCIRun to other software libraries and applications. Specific examples of

such mechanisms include the simplest step of integrating at the linker level the libraries that have interfaces suitable to direct access from SCIRun, as we have implemented for the Teem library (teem.sourceforge.net) and the BLAS library (www.netlib.org/blas/). A more complex linkage was necessary to interact with Matlab (www.mathworks.com), in which we used a Unix socket based interface to send data and commands from SCIRun to Matlab and receive results back. A further approach to linking to programs that run independently of SCIRun is to share data through a file or database, a strategy we implemented to communicate with the GENESIS nerve cell simulation system (http://www.genesis-sim.org). The most important finding of all these projects is that discovering a way to bridge to other high quality software systems, while sometimes challenging, is almost always a more efficient and satisfying approach than re-creating the software within the host system.

3.4. Software engineering

Creating and implementing consistent software engineering practices in a university based development project remains an ongoing challenge for us and for many other groups. Factors that contribute to this challenge include the lack of training among academics—even computer scientists—in software engineering concepts and practice; encouraging contributions from students who are training to become specialists in scientific disciplines and thus poorly motivated to take the required additional time; and the restricted budget typically available in academic settings to create, maintain, and enforce sound software engineering techniques. We have attempted to address some of these obstacles by consolidating resources in order to recruit trained software engineers and to secure support from the leadership of the projects in order to make the compromises in apparent progress required to create stable, robust, dependable computer programs.

3.5. Data I/O and conversion

Once a user has open source software compiled and installed, the most frequent question in our experience with scientific users is how they can import their own data into the program. Thus creating flexible tools for this purpose is an essential component of achieving widespread use of new software. Our approach in the SCIRun/BioPSE project has been to develop converters between our internal file formats and simple text based formats that we document clearly. In this way, users can often treat the text format as a lowest common form and create their own conversion support to their local software. While reasonably robust, this approach is not really adequate and we are planning an alternative based on an XML description of data that SCIRun/BioPSE can use to execute file conversion. By encapsulating the file format description in the XML (or some derivative) the software that manages data conversion can be stable and the customization burden shifts to the markup language which allows for faster iterations and better re-use of previous descriptions.

3.6. User support

Academic settings are typically much less equipped to manage user support than the private sector. However, the open source software movement has created a different–and often more forgiving—set of expectations with regard to user support. Dedicating valuable resources to user support is often even more difficult than the situation with regard to software engineering as academic institutions offer little support for positions and career paths in the related areas of technical writing and customer support. Fortunately, however, many users of open source software will tolerate a lower standard of support than would be acceptable in a commercial setting. Even more important, the user community of open source software is often prepared to provide mutual support through mailing lists and user group sessions at conferences or even whole meetings dedicated to using and developing a software package. A sign of vitality in any new software system is the first time that questions posed to the users' mailing list receive answers from other users before the developers of the software have a chance to reply. In the SCIRun/BioPSE project, we have used a combination of extensive documentation as well as email based support lists and annual workshops to educate and support users. We have used the software in courses offered to biomedical engineers and have conducted tutorial workshops at international conferences. Most useful in our experience have been the online tutorials we have developed for the entry level user.

4. REFERENCES

[1] S.G. Parker and C.R. Johnson, "SCIRun: a scientific programming environment for computational steering," in *Proc ACM IEEE Supercomputing Conf*. 1995, vol. 2, pp. 1419–1439, IEEE, Los Alamitos, CA.

[2] S.G. Parker, D.M. Beazley, and C.R. Johnson, "Computational steering software systems and strategies," *IEEE Computational Science and Engineering*, vol. 4, no. 4, pp. 50–59, 1997.

[3] J.D. de St. Germain, J. McCorquodale, S.G. Parker, and C.R. Johnson, "Uintah: A massively parallel problem solving environment," in *Ninth IEEE International Symposium on High Performance andDistributed Computing*. Nov 2000, pp. 33–41, IEEE, Piscataway, NJ.