Journal of Computational Science 52 (2021) 101217

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Translational computer science at the scientific computing and imaging institute

Chris Johnson [1]

Faculty Member and Founding Director, Scientific Computing and Imaging Institute, Distinguished Professor, School of Computing, University of Utah, United States

ARTICLE INFO

Keyword:
Translational computer science
Open source software development
Multidisciplinary research

ABSTRACT

The Scientific Computing and Imaging (SCI) Institute at the University of Utah evolved from the SCI research group, started in 1994 by Professors Chris Johnson and Rob

MacLeod. Over time, research centers funded by the National Institutes of Health, Department of Energy, and State of Utah significantly spurred growth, and SCI became a permanent interdisciplinary research institute in 2000. The SCI Institute is now home to more than 150 faculty, students, and staff.

The history of the SCI Institute is underpinned by a culture of multidisciplinary, collaborative research, which led to its emergence as an internationally recognized leader in the development and use of visualization, scientific computing, and image analysis research to solve important problems in a broad range of domains in biomedicine, science, and engineering. A particular hallmark of SCI Institute research is the creation of open source software systems, including the SCIRun scientific problem-solving environment, Seg3D, ImageVis3D, Uintah, ViSUS, Nektar++, VisTrails, FluoRender, and FEBio. At this point, the SCI Institute has made more than 50 software packages broadly available to the scientific community under open-source licensing and supports them through web pages, documentation, and user groups.

While the vast majority of academic research software is written and maintained by graduate students, the SCI Institute employs several professional software developers to help create, maintain, and document robust, tested, well-engineered open source software. The story of how and why we worked, and often struggled, to make professional software engineers an integral part of an academic research institute is crucial to the larger story of the SCI Institute's success in translational computer science (TCS).

1. Introduction

The Scientific Computing and Imaging (SCI) Institute at the University of Utah evolved from the SCI research group, started in 1994 by Professors Chris Johnson and Rob MacLeod [1,2]. Over time, research centers funded by the National Institutes of Health, Department of Energy, and State of Utah significantly spurred growth, and SCI became a permanent interdisciplinary research institute in 2000. The SCI Institute is now home to more than 150 faculty, students, and staff.

The history of the SCI Institute [2] is underpinned by a culture of multidisciplinary, collaborative research [3], which led to its emergence as an internationally recognized leader in the development and use of visualization, scientific computing, and image analysis research to solve important problems in a broad range of domains in biomedicine, science, and engineering. A particular hallmark of SCI Institute research is the creation of open source software systems [4], including the SCIRun scientific problem-solving environment, Seg3D, ImageVis3D, Uintah, ViSUS, Nektar++, VisTrails, FluoRender, and FEBio. At this point, the SCI Institute has made more than 50 software packages broadly available to the scientific community under open-source licensing and supports them through web pages, documentation, and user groups.

While the vast majority of academic research software is written and maintained by graduate students, the SCI Institute employs several professional software developers to help create, maintain, and document robust, tested, well-engineered open source software. The story of how and why we worked, and often struggled, to make professional software engineers an integral part of an academic research institute is crucial to the larger story of the SCI Institute's success in translational computer science (TCS) [25].

E-mail address: crj@sci.utah.edu.
[1] www.sci.utah.edu.

https://doi.org/10.1016/j.jocs.2020.101217
Received 17 August 2020; Received in revised form 1 September 2020; Accepted 2 September 2020
Available online 10 September 2020

2. Background

The SCIRun software system began in the early 1990s as a research project in computational steering and integrated problem solving environments [5,6]. As it evolved, SCIRun research and software development branched in two different directions associated with two very different research centers. In 1997, the Center for Accidental Fires and Explosions (C-SAFE) [9], created through the U.S. Department of Energy's Advanced Simulation and Computing Initiative (ASCI), comprised an interdisciplinary team of researchers in chemical engineering, mechanical engineering, chemistry, mathematics, and computer science. The overall goal of C-SAFE was to provide state-of-the-art, science-based tools for the numerical simulation of accidental fires and explosions, especially within the context of safely handling and storing highly flammable materials. The objective of C-SAFE was to provide an accurate, scalable software system within which fundamental chemistry and engineering physics are fully coupled with non-linear solvers, optimization, computational steering, visualization and experimental data verification. The C-SAFE team performed research in large-scale simulations of complex physical phenomena including reacting flows, material properties, multi-material interactions, and atomic-level chemistry. For more than a decade, the C-SAFE team performed pioneering work in the field of parallel computing, software frameworks, and visualization. C-SAFE scientists also performed research into large eddy simulations (LES) of reacting flows, immense combustion simulations, heat transfer studies, validation and verification with uncertainty quantification of simulation results, methods for modeling radiation in complex fire simulations, expansion and validation of the material point method (MPM), advanced chemical models of soot formation and deposition, and composite material modeling.

The NIH Center for Bioelectric Field Modeling, Simulation, and Visualization launched in 1999 [10] within the NIH National Center for Research Resources. During the first five years, the Center focused on creating an extensible, scalable, scientific problem-solving environment (PSE) and on developing corresponding research to solve real-world problems relating to bioelectric fields. To accomplish this goal, we conducted research and development in advanced modeling, simulation, and visualization methods for solving bioelectric field problems; we also created BioPSE. An extension to the existing SCIRun software system, BioPSE was a modular, extensible, integrated software problem-solving environment for bioelectric field problems. While the SCIRun software supports interaction among the modeling, computation, and visualization phases of bioelectric field simulation, BioPSE provided specific extensions for the Center. In 2005, the Center was renamed the Center for Integrative Biomedical Computing (CIBC) and transitioned to the Biomedical Technology Research Resource program within NIH NIGMS. At this time, the Center reengineered SCIRun to separate the underlying filters or modules from the dataflow interface. The Center also began integrating into its tools third-party packages like the Insight Toolkit (ITK), in turn assisting in the latest generation of Center applications, such as map3d, ImageVis3D, Seg3D, BioMesh3D, and ShapeWorks.

For decades, applied computing researchers have created software to solve science, engineering, and medical research problems. In the current academic environment, this software is usually created by graduate students and postdoctoral researchers. Needless to say, among this group the range of software engineering expertise varies greatly. Beyond this, because software is created within the context of very specific individual projects, in our experience the majority of scientific research software addresses a specific short term goal such as a paper or demo rather than being designed for long term use or more general distribution. Within the proposals to create the C-SAFE and CIBC centers, we committed to producing open source software that would be both available to and usable by other researchers. In making that commitment, we underestimated the difficulty of the task. We knew we had software that worked well for us, so we assumed we could just make it available for download with some instructions on how to install it and use it, and soon we would have thousands of happy users. We soon learned what anyone who has successfully transitioned research software to well designed, well-engineered, tested, documented, and sustainable software already knows: the task is tremendously challenging. For us, addressing it involved integrating professional software engineers into an academic research environment. Though such integration raises its own challenges, we have found it to be well worth the effort, as these software engineers continue to be key to many of our research projects.

Once we employed software developers, we discovered that incorporating them into a research group had its own challenges. First, the goals of software developers working to create stable builds were often in tension with those of graduate students, postdocs, and faculty who constantly required and worked to create new research features. Second, professional software developers are in high demand throughout industry, where there are more open positions for software engineers than there are qualified people to fill them. As such, talented software developers command high salaries compared to graduate students and postdocs; some senior software developers command salaries that are higher even than those of junior faculty. While a few research agencies have sometimes funded professional software developers, this is not the norm. Even when a research group can obtain funding for good developers, they find it difficult to compete successfully against industry to hire and retain them.

3. Translation process

Both the C-SAFE and CIBC Centers had multiple computer science research aims including parallel adaptive mesh refinement, material point method, parallel rendering, shape modeling, image segmentation, uncertainty visualization, that needed to be translated from research code into usable software. Because there is a significant difference between the research proof-of-concept software produced by labs for internal use and the robust applications that users expect when they download a software product from the web, we proposed within the CIBC the aim of creating Translational Software, to transform new algorithms developed in Center imaging, simulation, and visualization research cores into stable, available, documented software projects. Once the potential of a new technology has been established, significant additional software engineering efforts are required to generate a robust application that interfaces well with the rest of the framework and runs stably on multiple computer platforms. The additional software engineering includes cleaning up the application's source code, refactoring the application for integration with the Center's standard framework components, documentation of the application's functionality, and a rigorous set of white- and black-box tests to be constructed around the application. Although these software engineering steps do not necessarily add any novel functionality to the programs, they are necessary to ensure the sustainability of the software outside of the Center (Fig. 1)

3.1. Integrating software engineers within the research environment

It was challenging to build a robust software system that we could maintain and distribute (i.e. real software engineering), while at the same time supporting graduate students using it for research projects. One solution was to build Packages. The software engineers maintained the Datatypes, Modules, and Command-Line utilities in the main SCIRun and BioPSE Packages, and the graduate students used a combination of Modules from those main Packages, plus Modules/Datatypes/Utilities that they added in their own Packages. We also relied heavily on templates, so people could extend capabilities without breaking other people's code. The downside of templates was that they required long compile times, which we addressed with on-the-fly compilation. After the usual early "build everything yourself" phase, another software sustainability solution was to move to standards such CMake, CTest, and CDash.

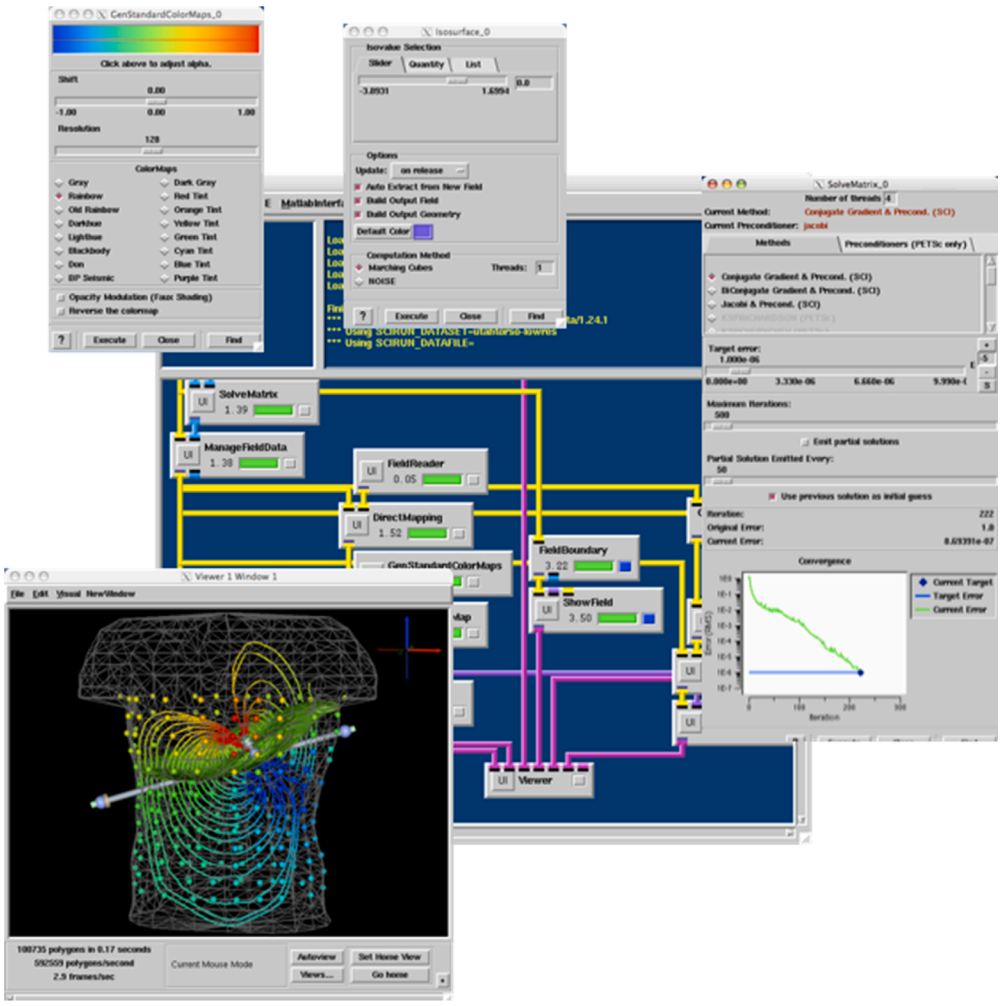

Fig. 1. (a) The SCIRun PSE showing the network, module interfaces and visualization window. (b) The BioPSE application showing the results of a cardiac bioelectric field simulation within the Utah torso model [28,29]. The graph on the right side shows the convergence of the iterative solution technique.

While maintaining both stable and research software trees, moving graduate student research code into the stable software builds remains a challenge. No matter the level of encouragement, good software engineering practices are still difficult to enforce among computational science graduate students and postdocs, who don't easily embrace the rigors of the software engineering culture. One of our most gifted graduate students announced that "great software documents itself". He later retracted this statement after not being able to understand some of the many thousands of lines of optimized C++ code he himself had written over the years.

One way we addressed this developer/researcher cultural mismatch was to hire a technical software manager who both has a Ph.D. and really understands software, so they have a solid foot in each "camp" and can communicate effectively at highly technical levels between developers and researchers, often acting as a translator. The technical manager has proved an essential person in helping to better understand expectations and workload and also to project software personnel needs for research projects.

An important feature of the CIBC structure was the requirement that the Center have multiple Driving Biomedical Projects (DBPs). Driving Biomedical Projects are biomedical research test-beds that allow Center investigators to test nascent technologies in the context of challenging problems in basic, translational, and clinical research, while providing biomedical researchers with the earliest possible access to these emerging tools. A deep understanding of needs and opportunities in the relevant areas of biomedical research is an essential prerequisite for all technology development. In the Center, this understanding is most clearly expressed through successful engagement of those researchers best positioned to benefit from early access to emerging tools. The Center is expected to develop new technologies that will significantly impact a broad community of biomedical researchers, and through leadership within the relevant communities, support the integration of those technologies into the larger context of the relevant field. To help understand the impact of our software, as part of our annual review by our external advisory board, we were expected to produce a summary of software download numbers and papers that cited the use of our software. For example, the CIBC's Seg3D software has had more than 14,000 downloads and has been cited more than 200 times in published papers.

To describe the translation from DBP partner to the larger biomedical research community, we coined the phrase "Cone of Influence" (Fig. 2) This Center structure required us to work very closely with our DBP partners. Examples include DBP partner Natalia Trayonova, the Murray B. Sachs Professor in the Department of Biomedical Engineering and the Institute for Computational Medicine at Johns Hopkins University and a leader in the application of multiscale simulation to cardiac rhythm disturbances. She and her large group of investigators implement computational and simulation approaches in a clinical setting. We have worked with Professor Trayanova both in the efficient generation of patient specific models of the human heart in patients with arrhythmias and in shape-based analysis of cardiac structure in normal subjects and patients with structural heart disease [23,24]. Another example is DBP partner Gabrielle Kardon, the Director of the Kardon Laboratory in the Department of Human Genetics at the University of Utah. She uses multiphoton confocal microscopy for research into musculoskeletal development and regeneration. Professor Kardon has worked with the CIBC on the development of the FluoRender visualization software system to understand the molecular mechanisms and cellular interactions regulating musculoskeletal development and regeneration [20]. While Professor Kardon has used FluoRender with musculoskeletal development applications, FluoRender has now been used for a large number of diverse applications from three-dimensional reconstruction of *Drosophila* neural circuits [21] to three-dimensional imaging of plant organs [22]. The DBP partnerships are good examples of TCS, exhibiting the technical development within the CIBC, the back and forth interaction with our DBP researchers, and, in many cases, the extension of the research and software development to the broader research community for use in a larger application space.

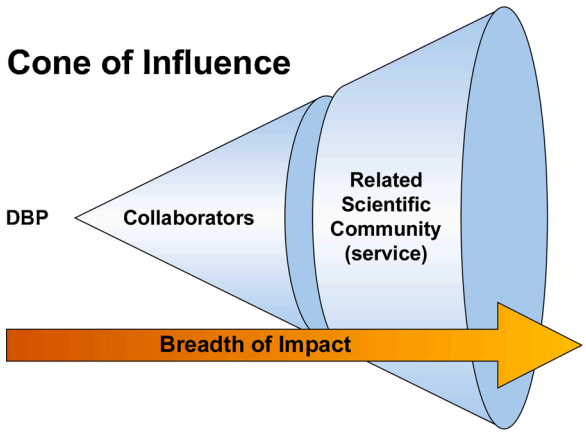

Fig. 2. Cone of influence. Translating research and development with our driving biomedical project (DBP) partners to the larger biomedical research community.

The Seg3D image segmentation system [13] is a good example of the TCS workflow in which we created a new software system from what we learned from the process of working with our DBP research partners, while at the same time thinking ahead to a system that could have broad research impact. Seg3D is a volume image processing and segmentation tool that enables researchers to quickly and easily explore and process their 2D and 3D medical imaging data. Seg3D is both simple to use and powerful, combining the wealth and sophistication of tools from the ITK [14] library with simple interfaces and manual segmentation tools. This design makes Seg3D a versatile tool for researchers working with three-dimensional imaging data within multiple modeling and simulation domains. Because of its sophisticated design, simple and intuitive user interface, and ability to manipulate imaging data, it is often the first tool that researchers use on their data in the image-based modeling pipeline that we have developed within the CIBC. Additionally, the tools in Seg3D provide a convenient platform for correcting outputs of other automatic segmentation pipelines, which often achieve incomplete results that require additional manual editing. While Seg3D is designed to be simple to use, advanced tools such as provenance and the Python interface allow it to be scripted and used in automatic segmentation workflows. Seg3D was initially developed with our research clinical cardiology DBP partners, but soon found broader application in a wide variety of applications including biomechanics, orthopedics, dentistry, plant biology, entomology, and even archeology (Figs. 3, 4 and 5)

3.2. *Challenges of funding, hiring, and retaining*

As previously noted, funding, hiring, and retaining good software engineers is an ongoing challenge. For computational science and engineering researchers, obtaining research support for graduate students and postdocs is standard, but most single PI grants do not have large enough budgets for supporting a professional software engineer [11]. We continue to tackle this challenge in multiple ways. Our first software engineers were recruited as part of the multidisciplinary research centers, C-SAFE and CIBC. As the SCI Institute faculty saw the benefits of these developers, we worked together to hire additional software engineers spread across multiple grants, finding that having one developer working on at most two research projects is ideal. Having multiple software developers that could be shared across multiple projects has proved to be a great way to increase and share developer expertise. We have software engineers with expertise in databases, graphics, and multiple programming languages, as well as experiences with a wide variety of architectures. Having the ability to bring in specific software

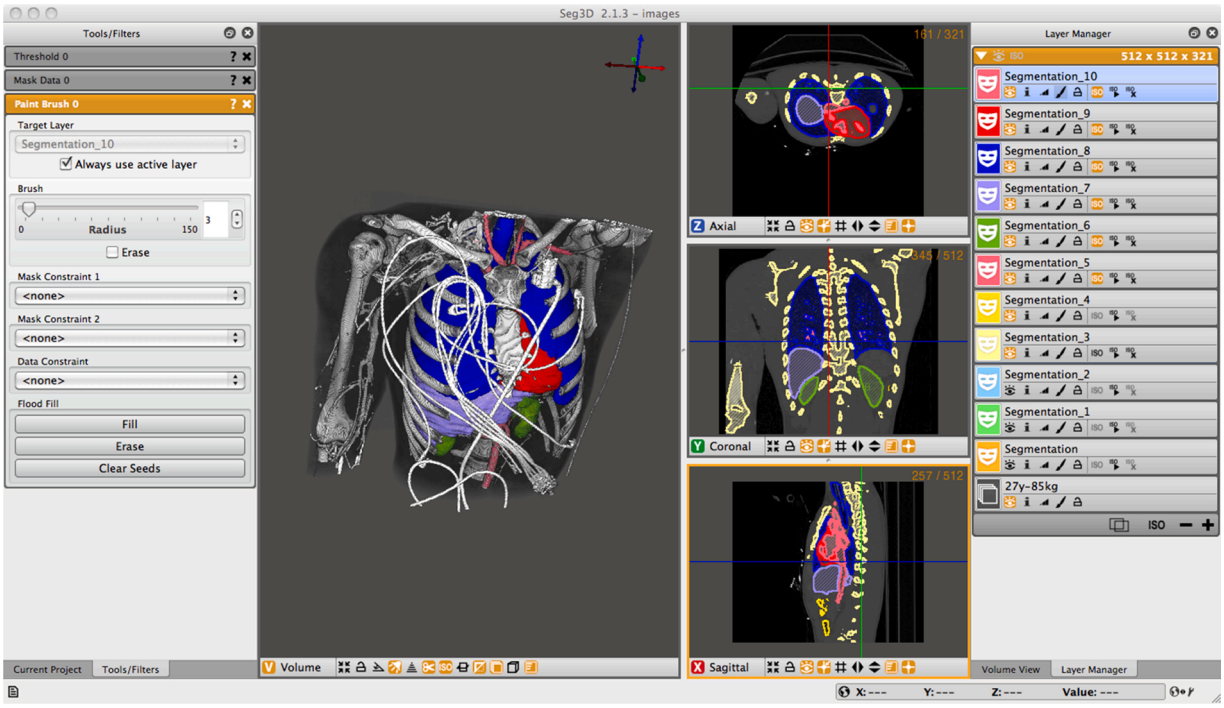

Fig. 3. Visualization and segmentation from a CT scan of a healthy adult using Seg3D [13]. This segmentation was used to generated patient specific models with many applications in cardiac electrophysiology, including predicting defibrillator efficacy, simulating cardiac potentials throughout the torso, and predicting cardiac activity from torso measurements. The three panels on the right show the axial, coronal, and sagittal views.

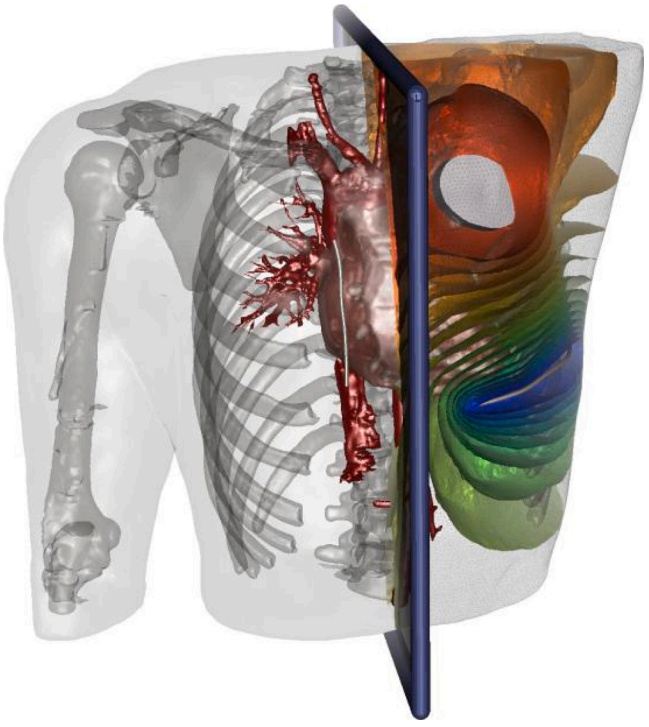

Fig. 4. Visualization of a three-dimensional finite element model created from patient images. First, a segmentation was done using Seg3D. Second, a three-dimensional mesh was created using BioMesh. Third, a large-scale finite element simulation was performed using BioPSE. Finally, the model and simulation results showing multiple voltage isosurfaces were visualized using SCIRun.

engineering expertise has been a powerful asset for some larger multi-disciplinary projects and even for smaller research projects that need a few months of graphics or database or user interface design.

Given the demand for good software engineers, we continue to seek creative ways to hire and retain developers. Because we are an academic research institute, we understand that we will not be able to compete with industry salaries, especially for more senior software engineers. We often hire recently graduated computer scientists with the offer to expose them to a wide variety of challenging software projects, thus expanding their skill set and expertise and making them more valuable to industry. Getting undergraduate computer science students involved in programming for research projects has been another way we recruit software engineers. As our software engineers gain experience and expertise, they become increasingly interesting to industry, so we expect to have many of our software engineers recruited away to higher paying industry positions. One way we retain software engineers is to support their continued education through specialized courses and programs and graduate degrees; as a result, we have had multiple software engineers complete MS degree programs. In the UK, the EPSRC has recognized the importance of investing in software development [15] and has recently establish a Research Software Engineering Fellowship [16] to aid in establishing this area as a career path within academic research universities. We hope this idea will spread globally.

3.3. Additional translational efforts

Two additional research software engineering translational efforts include helping to set up a Software Development Center at the University of Utah [19] and our new collaboration with Kitware (www.kitware.com) to transition the CIBC's open source software into well crafted, validated computer code, with community-supported and sustainable support for both users and future maintainers of the code base.

The Software Development Center (SDC) was a joint effort between the University of Utah's Technology Commercialization Office (TCO), which manages all intellectual property on the University of Utah campus, and the SCI Institute. The Software Development Center goals

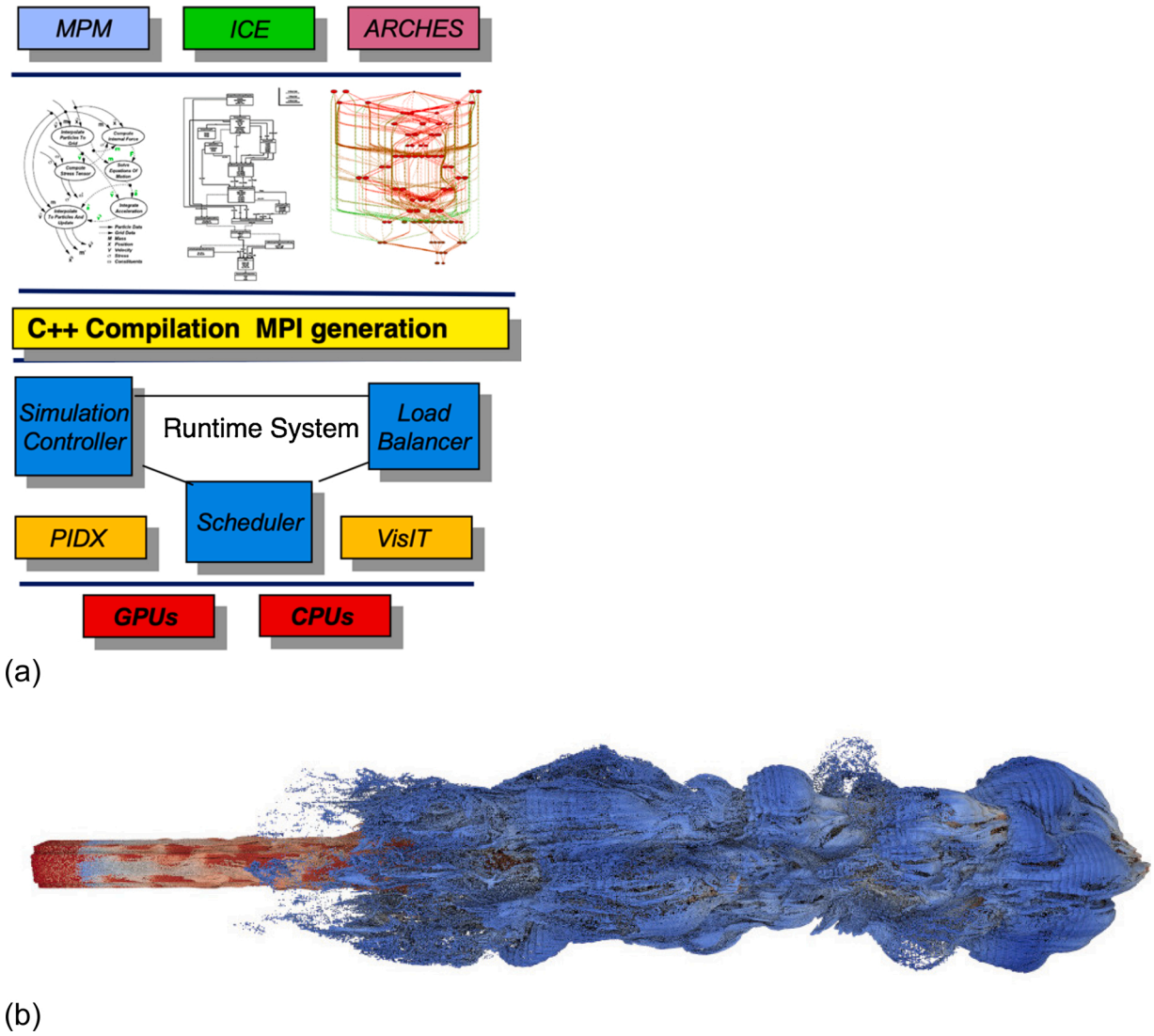

Fig. 5. (a) Overview of Uintah software architecture [17]. The top row shows the simulation systems. Arches is a three dimensional, Large Eddy Simulation (LES) code developed at the University of Utah. Arches is used to simulate heat, mass, and momentum transport in reacting flows [30]. The Material Point Method (MPM) is a computationally effective particle method for solving solid mechanics problems involving large deformations [31] and ICE is an implicit continuous-fluid Eulerian solver that can be coupled to Uintah's particle-based solver for solids (MPM) for fluid-structure interaction problems. PIDX is a scalable and tunable adaptive resolution parallel I/O framework [32] and VisIt is an open source, interactive, scalable, visualization, animation and analysis tool [33]. (b) Uintah simulation of coal particles being injected into a large-scale boiler.

Collaborations

were to:

- Help researchers and entrepreneurs develop software.
- Create a central repository for campus software projects.
- Create startup companies based on software projects.
- Commercialize software developed on campus.
- Provide students with professional software experience.
- Develop a talent pool for the state's software industry.
- Make software developed on campus available to the public.

In the formative stages, the SDC was led by Greg Jones, then Associate Director of the SCI Institute, who oversaw multiple professional software engineers and coordinated with University faculty and the TCO. The senior lead software engineering staff had PhDs in Computer Science and multiple years of experience. Their experience and technical expertise proved crucial in their being able to interact effectively with faculty from different disciplines in science, engineering, and medicine to understand how to translate research codes and ideas into well software engineered systems. The SDC helped develop several software systems and worked with a number of University start-up companies on their software.

One of the positive translational consequences of both the SDC and having multiple software engineers at the SCI Institute is the increased interaction of graduate students with professional software developers. One effect is that graduate students are informally mentored about professional software practices and learn both to create high quality software and also what is involved in creating new features and maintaining software beyond their graduate studies. Translating our software to a broad research community allows our graduate students to learn valuable software engineering skills that help them in their future research and industry positions.

The newly established CIBC software transition plan with Kitware Inc. is to achieve long term sustainability of the software and data resources that the CIBC has produced. The goal of this plan is to complete the conversion of all the technical products achieved over the lifetime of the Center into well crafted, validated computer code with necessary

support for both users and future maintainers of the code base. The conversion will require the assistance of a professional software house that is familiar with the necessary steps and with the technical domain of our center, Kitware Inc. Kitware is a company with a 20-year history of assisting biomedical researchers to develop highly complex software packages and libraries, including VTK, ITK, 3D Slicer, and Paraview. They have also developed highly flexible systems to build, deploy, and maintain software and are well versed in the practical aspects of supporting communities of users of—and contributors to—scientific software projects.

The main innovations of this project include the transition of software from an academic research institute to an open-source commercial setting and the leverage of an active user community to maintain and enhance the capabilities of the software. Past partnerships between public academic and private companies like Kitware have been in place from the start of the development process (*e.g.,* 3D Slicer or ITK), while, in our case, the CIBC has developed a rich and advanced set of software engineering practices informed by, but separate from, other entities. The proposed project involves seven different software packages to be evaluated then transitioned to the Kitware infrastructure. The second innovation will be to leverage a user community—more correctly, a set of domain specific user communities—to provide ongoing support for the software tools. The impact of the open-source software movement is well documented, and recent examples of crowd-sourcing to process scientific data have motivated us to propose such an approach. Success in either of these innovations will open new avenues for extracting valuable, often publicly-funded software from academic centers and making it freely available to the scientific community. From a software engineering perspective, we have also devised a formalized sustainability matrix to characterize the state of each software system in terms of long-term sustainability [27]. We will use this scoring approach to deploy long-lasting, continually impactful software systems. Seg3D is one of the first CIBC software systems that we will work with Kitware to transition to their software infrastructure and sustainability protocols.

4. Impact and lessons learned

Investing in professional software engineers as team members within our academic research institute has yielded multiple long-term benefits. These engineers have helped to create and maintain more than 50 open-source software packages that are broadly available to the scientific community and are supported by web pages, documentation, and user groups. Examples of TCS methodologies and impact were highlighted by experiences with two large, long running, multidisciplinary research centers, the CIBC and C-SAFE, that integrated cutting edge research with multiple software projects.

The overarching goal of CIBC was to advance the state of practice in biomedical computing and its applications both to biomedical science and to the translation of this science to clinical practice. We achieved this goal by making advanced computational tools, tailored to the specific domains of image-based modeling, simulation, estimation, and visualization, accessible to scientists, engineers, and physicians, by releasing readily usable, open-source software. We worked closely with these users of our software, providing advice, technical support, workshops, and education to enhance their success with the tools we provide.

The origins of our success in developing widely-used software tools lie in a set of strategies for algorithm research and software development. One strategy has been the production of software tools with low barriers to entry. This entails the release of documented, tested, complete applications that do not entail learning new programming languages or complex, architecture-specific build environments. This strategy is reflected in the complete merging of SCIRun and BioPSE, problem-solving environments.

Another important TCS strategy for software development is to make our tools relevant for the broader community by building and testing them in close collaboration with specific, high-profile biomedical groups, our DBP research partners. These practicing scientists, engineers and physicians educated us to achieve an appropriate level of generality to impact the largest group possible at the same time that our close collaborations with them help us to define and solve specific, relevant biomedical problems. These collaborators also inform us about other tools that investigators are using, thus helping us to ensure that our software fills a clear need in the community and remains at (or ahead of) the state of the art. And, since the proof of the effectiveness of the Center is in the success of our users, this collaboration strategy and the high profile of our collaborators has also proven to be a mechanism for dissemination of tools.

A key success of C-SAFE was that the physical science research was greatly augmented by the underlying software framework Uintah [7,8,17], which had its origins in SCIRun. The TCS strategy of technical innovation within our research team, working with key application scientists and engineering and broadening these results to larger research communities, proved successful. Because of the utility of the software, which scales to hundreds of thousands of processors and is used for many large-scale simulation research applications, the development of the Uintah computational framework continues through a number of additional research funding sources, and the software is constantly refined and updated. While developed primarily to run fire/container interaction simulations, Uintah has been straightforwardly applied to many other diverse applications, including simulations of blood vessel growth, foam micro-structure, human torso dynamics, industrial flares, vehicle armor plating, and oil drilling applications.

In addition, the parallel ray tracing visualization research [12] started in C-SAFE yielded a spinoff company, RayScale, that was acquired by NVIDIA. Pieces of that technology grew along with other NVIDIA ray tracing technology to become their RTX platform [18].

While TCS has the potential to accelerate the impact of computer and computational science research, it has largely been implemented by individuals and groups in ad hoc ways, and it is not usually recognized as a specific methodology. To promote TCS and learn to apply the TCS workflow, we will need to establish new funding and reward mechanisms that encourage, support and realize TCS structures. I could imagine creating a small number of distributed TCS centers to act as “expeditions to the future,” as described in the 1999 PITAC report [26].

Author statement

Chris Johnson – single author paper.

Declaration of Competing Interest

The authors reported no declarations of interest.

Acknowledgments

The author acknowledges current research support provided by the Intel Graphics and Visualization Institutes of XeLLENCE, the National Institute of General Medical Sciences of the National Institutes of Health under grant numbers P41 GM103545 and R24 GM136986 and the Department of Energy under grant number DE-FE0031880.

References

[1] Scientific Computing and Imaging (SCI) Institute at the University of Utah, 2020. http://www.sci.utah.edu.
[2] History of the Scientific Computing and Imaging Institute. Evert L. Cooley Collection, Marriott Library, University of Utah, 2018. http://www.sci.utah.edu/images/Research/SCI_History_Cooley.pdf.
[3] B. Shneiderman, The New ABCs of Research: Achieving Breakthrough Collaborations, Case Study 3.6, Oxford University Press, 2016. http://www.cs.umd.edu/hcil/newabcs/.
[4] SCI Institute Software, 2020. http://www.sci.utah.edu/sci-software.html.

[5] C.R. Johnson, S.G. Parker, A computational steering model applied to problems in medicine, in: Supercomputing, 94, IEEE Press, 1994, pp. 540–549.
[6] S.G. Parker, D.M. Weinstein, C.R. Johnson, The SCIRun computational steering software system, in: E. Arge, A.M. Bruaset, H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhauser Press, Boston, 1997, pp. 1–40.
[7] J.D. de St. Germain, J. McCorquodale, S.G. Parker, C.R. Johnson, Uintah: a massively parallel problem solving environment. The Ninth IEEE International Symposium on High Performance and Distributed Computing, IEEE, Piscataway, NJ, 2000, pp. 33–41. Nov.
[8] M. Berzins, J. Beckvermit, T. Harman, A. Bezdjian, A. Humphrey, Q. Meng, J. Schmidt, C. Wight, Extending the uintah framework through the petascale modeling of detonation in arrays of high explosive devices, SIAM J. Sci. Comput. 38 (5) (2016) S101–S122, https://doi.org/10.1137/15M1023270.
[9] Center for the Simulation of Accidental Fires and Explosions (C-SAFE), 2020. http://csafe.sci.utah.edu.
[10] Center for Integrative Biomedical Computing, 2020. http://www.sci.utah.edu/cibc.
[11] U. Rüde, K. Willcox, L.C. McInnes, H. De Sterck, G. Biros, H. Bungartz, J. Corones, E. Cramer, J. Crowley, O. Ghattas, M. Gunzburger, M. Hanke, R. Harrison, M. Heroux, J. Hesthaven, P. Jimack, C. Johnson, K.E. Jordan, D.E. Keyes, R. Krause, V. Kumar, S. Mayer, J. Meza, K.M. Mørken, J.T. Oden, L. Petzold, P. Raghavan, S.M. Shontz, A. Trefethen, P. Turner, V. Voevodin, B. Wohlmuth, C. S. Woodward, Research and education in computational science and engineering, Siam Rev. 60 (3) (2018) 707–754.
[12] S.G. Parker, W. Martin, P.-P. Sloan, P. Shirley, B. Smits, C.D. Hansen, Interactive ray tracing, Symposium on Interactive 3D Graphics: Interactive 3D (1999) 119–126. April 26-28.
[13] Seg3D Image Segmentation Software, 2020. http://www.sci.utah.edu/software/seg3d.html.
[14] ITK: The Insight Toolkit, 2020. https://itk.org.
[15] EPSRC: Software As an Infrastructure, 2020. https://epsrc.ukri.org/newsevents/pubs/software-as-an-infrastructure/.
[16] Research Software Engineers Fellows, 2020. https://epsrc.ukri.org/funding/calls/rsefellowships/.
[17] Uintah Software System, 2020. http://uintah.utah.edu.
[18] NVIDIA RTX Platform, 2020. https://developer.nvidia.com/rtx.
[19] University of Utah Software Development Center, 2020. https://archive.unews.utah.edu/news_releases/software-development-center-opens-doors/.
[20] Y. Wan, H. Otsuna, H.A. Holman, B. Bagley, M. Ito, A.K. Lewis, M. Colasanto, G. Kardon, K. Ito, C. Hansen, FluoRender: joint freehand segmentation and visualization for many-channel fluorescence data analysis, BMC Bioinform. 18 (May 1) (2017). Springer Nature.
[21] K. Shinomiya, M. Ito, Recent progress in the 3D reconstruction of *Drosophila* neural circuits, in: A. Çelik, M. Wernet (Eds.), Decoding Neural Circuit Structure and Function, Springer, Cham, 2017, https://doi.org/10.1007/978-3-319-57363-2_3.
[22] Junko Hasegawa, Yuki Sakamoto, Satoru Nakagami, Mitsuhiro Aida, Shinichiro Sawa, Sachihiro Matsunaga, Three-dimensional imaging of plant organs using a simple and rapid transparency technique, Plant Cell Physiol. 57 (March 3) (2016) 462–472, https://doi.org/10.1093/pcp/pcw027.
[23] K.S. McDowell, S. Zahid, F. Vadakkumpadan, J. Blauer, R.S. MacLeod, N. Trayanova, Virtual electrophysiological study of Atrial Fibrillation in fibrotic remodeling, PLoS One 11 (5) (2016) e0156189, https://doi.org/10.1371/journal.pone.0156189.
[24] A. Prakosa, H.J. Arevalo, D. Deng, P.M. Boyle, P.P. Nikolov, H. Ashikaga, J. E. Blauer, E. Ghafoori, C.J. Park, R.C. Blake III, F.T. Han, R.S. MacLeod, H. R. Halperin, D.J. Callans, R. Ranjan, J. Chrispin, S. Nazarian, N.A. Trayanova, Personalized virtual-heart technology for guiding the ablation of infarct-related ventricular tachycardia, Nat. Biomed. Eng. 2 (2019) 732–740.
[25] D. Abramson, M. Parashar, Translational research in computer science, Computer 52 (09) (2019) 16–23, https://doi.org/10.1109/MC.2019.2925650.
[26] Information Technology Research: Investing in Our Future, President's Information Technology Advisory Committee (PITAC), 1999. February, https://www.nitrd.gov/pitac/report/pitac_report.pdf.
[27] G. Lami, L. Buglione, Measuring software sustainability from a process-centric perspective, in: 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement, Assisi, 2012, pp. 53–59.
[28] C.R. Johnson, Computational and numerical methods for bioelectric field problems, Crit. Rev. Biomed. Eng. 25 (1) (1997) 1–81.
[29] C.R. Johnson, R. MacLeod, S.G. Parker, D. Weinstein, Biomedical computing and visualization software environments, Commun. ACM 47 (11) (2004) 64–70.
[30] D. Sahasrabudhe, M. Berzins, Improving performance of the hypre iterative solver for uintah combustion codes on manycore architectures using MPI endpoints and kernel consolidation, in: Computational Science – ICCS 2020, 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part I, Springer International Publishing, 2020, pp. 175–190.
[31] M. Steffen, R.M. Kirby, M. Berzins, Analysis and reduction of quadrature errors in the material point method (MPM), Int. J. Numer. Methods Eng. 76 (6) (2008) 922–948.
[32] PIDX Parallel I/O Framework, 2020. https://www.sci.utah.edu/software/pidx.html.
[33] VisIt Visualization Software, 2020. https://wci.llnl.gov/simulation/computer-codes/visit/.

Chris R. Johnson is a Distinguished Professor of Computer Science and founding director of the Scientific Computing & Imaging (SCI) Institute at the University of Utah. He also holds faculty appointments in the Departments of Physics and Bioengineering. His research interests are in the areas of scientific computing and scientific visualization. In 1992, with Professor Rob MacLeod, Professor Johnson founded the SCI research group, now the SCI Institute, which has grown to employ over 150 faculty, staff and students. Professor Johnson serves on a number of international journal editorial and advisory boards to national and international research centers. He is a Fellow of AIMBE (2004), AAAS (2005), SIAM (2009), and IEEE (2014) and was recently inducted into the IEEE Visualization Academy (2019). He has received a number of awards including the NSF Presidential Faculty Fellow (PFF) award from President Clinton, a DOE Computational Science Award, the Governor's Medal for Science and Technology, the Utah Cyber Pioneer Award, the IEEE Visualization Career Award, IEEE IPDPS Charles Babbage Award, the IEEE Sidney Fernbach Award, and the Rosenblatt Prize.