2064
doi:10.1017/S1431927616011168
Microsc. Microanal. 22 (Suppl 3), 2016

Performing Quantitative Imaging Acquisition, Analysis and Visualization Using the Best of Open Source and Commercial Software Solutions.

Shailesh M. Shenoy[1]

[1.] Department of Anatomy & Structural Biology, Gruss Lipper Biophotonics Center, and Integrated Imaging Program, Albert Einstein College of Medicine, Bronx NY 10461, USA

A challenge in any imaging laboratory, especially one that uses modern techniques, is to achieve a sustainable and productive balance between using open source and commercial software to perform quantitative image acquisition, analysis and visualization. In addition to considering the expense of software licensing, one must consider factors such as the quality and usefulness of the software's support, training and documentation. Also, one must consider the reproducibility with which multiple people generate results using the same software to perform the same analysis, how one may distribute their methods to the community using the software and the potential for achieving automation to improve productivity.

As the tools for developing software become more accessible, scientists are increasingly applying their ingenuity to innovate not only sample preparation and instrumentation, but also to customize the software that is responsible for data acquisition, analysis and visualization. The ultimate control that a developer has when synthesizing code can lead to a precisely integrated solution that one can distribute easily and at no cost, to either the developer or end-user, online using resources such as GitHub [1]. Usually, a developer would like to achieve a level of abstraction between the solution and the actual hardware utilized by the researchers. For example, when authoring software to acquire data, it often would be desirable to abstract the image acquisition device so that the solution is not device or driver specific. If not, the end-users of the solution would be bound to use a configuration identical to that of the developer. For that reason and others, the best practice for developers is to use an open source standard as the baseline code from which to develop a solution. For example, one may use Micro-Manager for building an image acquisition solution [2] or ImageJ for building an image analysis function [3]. While the developer needs to conform to the programming conventions of the open source solution, the advantage of developing a solution using an open source standard is that the developer has a jump start on the process of making the solution. The authors of open source standards have deployed architectures that apply programming best-practices that lead to more manageable code. Also, the laboratory that develops a solution may choose to distribute the result of its development effort freely as a plugin that integrates with the open source software. The research community may then download and use the entire solution at will and at no cost. A particular challenge in this model for sharing developments is how to maintain a good set of documentation for the software and how to provide adequate training to end-users. Too often the strengths of an innovative software developer do not align well with the strengths necessary for providing clear and consistent documentation or training. Also, sustaining the project as an active development endeavor is often difficult as the primary developers move to work on other innovative projects.

Industry provides mature development environments that developers may use as a resource for prototyping and developing solutions using a library of pre-built resources that are well documented and supported by experts. For example, one may use LabVIEW for building and image acquisition solution [4] or MATLAB for building and image analysis tool [5]. The advantage of using these mature

platforms is that the developer can concentrate on providing the unique intellectual contribution to the field without having to reinvent portions of code that already exist in a mature form. These development platforms provide rich user interface options that make the end product easier to learn and use. The disadvantages are the costs associated with buying the software and maintaining the software licenses annually. Also, there are often technical and license restrictions that govern how the developer may distribute the final product, which in many cases leads to the end user needing to purchase a license for the development platform in order to use the solution.

Industry also provides countless feature-rich solutions that do not provide much flexibility for the end-user to innovate. These solutions provide value to the community as they offer a stable experience that is ubiquitous across a product line and over time as the product is actively developed to take advantage of features in new operating systems or to use new hardware. The advantage of this approach is that the commercial partner has taken the responsibility of assuring, for example, the proper integration of the software with the hardware. A potential hazard with using this approach is having to use proprietary file formats or not being able to share with others configuration files for software settings.

There are practical considerations for how a laboratory most efficiently achieves its goals with reasonable value. An emerging criterion for evaluating software is how a lab may document its methods for publication and distribution to others in an electronic fashion. The need to be able to communicate methods and findings concisely is becoming more important as data sharing becomes a mandate in academic research.

References:

[1] "Build Software Better, Together." GitHub. (2016) <https://github.com/>.
[2] "Micro-Manager Open Source Microscopy Software." Micro-Manager. (2016) <https://www.micro-manager.org/>.
[3] "ImageJ." ImageJ. (2016) <http://rsb.info.nih.gov/ij/>.
[4] "LabVIEW System Design Software." - National Instruments. (2016) <http://www.ni.com/labview/>.
[5] "MATLAB." - The Language of Technical Computing. (2016) <http://www.mathworks.com/products/matlab/>.
[6] The author acknowledges funding from the National Institutes of Health for project EB021236 awarded to RH Singer and project CA100324 awarded to J Condeelis and generous support of the Integrated Imaging Program by the EGL Charitable Foundation.