Software Sustainability in the Molecular Sciences

Theresa L. Windus, twindus@iasate.edu, Iowa State University and Ames Laboratory
T. Daniel Crawford, crawdad@vt.edu, Virginia Tech

The molecular sciences - including chemistry, materials, biophysics and biochemistry - have a long history of developing software to solve core scientific challenges. The field also has a long history of challenges to software sustainability. This whitepaper will discuss some of the software sustainability challenges and the opportunities/possible solutions that the Molecular Sciences Software Institute (MolSSI - https://molssi.org/) is working toward with the molecular sciences software development community.

MolSSI is an NSF funded project that is a nexus for science, education, and cooperation for the global computational molecular sciences community. Funded in 2016, the goals of MolSSI are to provide software expertise and infrastructure, education and training, and community engagement and leadership in molecular sciences software development. The fundamental purpose of the MolSSI is to serve and enhance the software development efforts of the broad field of computational molecular science.

Challenges:
The molecular sciences software ranges from small utility programs used to manipulate input and output to analysis programs to libraries that provide a particular functionality to monolithic scientific codes that are capable of many types of simulations. Each code is usually developed and optimized with a particular computing environment in mind (such as on a laptop or workstation, on a small cluster, or on the emerging exascale and quantum computers). *In toto* this software runs on the complete suite of computational hardware. In addition, the software is developed with many different types of languages (Fortran, Python, C++, C, and scripting languages, for example) and dependencies on different math libraries and compiling environments.

The diversity and evolving set of computational hardware and software stacks lead to significant sustainability challenges. Molecular computational scientists, as with most software developers, want their software to be portable and run on multiple platforms (as appropriate). This is one reason why most developers are happy to use math libraries that are optimized by the hardware and software vendors to perform well on a particular platform. Most of these libraries have common abstract programming interfaces (APIs) that allow the developer to link in different math libraries without having to make changes to the application software. Unfortunately, APIs and shared data formats are not common for most of the rest of the molecular sciences software ecosystem. This means that sharing of software is not as easy as it should be. Also, new or faster algorithms for a particular hardware are usually not adopted by other codes. Therefore, developers end up optimizing similar application software on the same hardware and software stacks. This extra work translates into more resources being needed across the community to sustain the software.

Much of the early software in the field was developed in an academic setting where little or no software engineering practices were used. To be fair, the wealth of software engineering practices in vogue today were not even developed or taught at the time of these early programs. In addition, there was generally no formal training of molecular scientists in the computer sciences including use of languages and data structures. Even today, graduate degrees in the molecular sciences generally require a minimum of courses and adding additional courses is often discouraged in lieu of more time spent on research. Undergraduate curricula in the molecular sciences already include a significant amount of course work and adding additional requirements can be difficult, especially if the faculty are not convinced that computational training is necessary. Unfortunately, even with the advances made in computer science and engineering and changes in the social environment in some molecular science departments, most of the computer science training for computational molecular scientists is still *ad hoc*; although, the wealth of online materials and courses makes obtaining this knowledge much easier than it used to be.

Sustainability is as much a socio-economic problem as a technical one; thus, there is a supply and a demand side to software sustainability. On the supply side, the cost model for the program needs to be considered, as well as the commitment of the developer community. If the software is open source, this includes the cost of the developers for the initial development and incentives and appropriate licensing models that facilitate external developer communities. Of importance here are low barriers to entry - ease of development, documentation, communication among the software developers, etc. It must be understood that developing this community can take time and effort, and therefore, expense. In addition the software development must align with the user with the user needs. Consistent and frequent communication with potential and existing users is necessary to ensure that the demand will be high. Factors that determine the engagement of users include tangible metrics such as software quality, as well as intangible metrics such as external user community's confidence in the code and process. The software development must be driven by maximum quality and community confidence in the code. For some software a cost recovery model will require providing support of, and services around, software as opposed to selling per use licenses. In addition, it must be understood that not all software should be sustained. Natural attrition of software is something we should expect in an ever changing socio-economic environment.

Another social challenge in the molecular sciences is that academic software is usually developed only as part of the process of producing simulations or verifying theoretical developments - to get the physics right - for publications in journals and toward advancement in a graduate degree. In other words, the software is often not the primary focus of the development of the software and is a vehicle used towards a publication - the primary metric still used in most science fields. This can lead to developers skipping the necessary upfront cost for clear and effective design decisions, taking shortcuts in the development to "just get the software working", skipping testing for end cases, and putting off documentation (that then never appears) - as examples. This can make the software difficult to maintain and, therefore,

difficult to sustain. Often, those developers who do take the time to produce well thought out, hardened software produce fewer publications. Although the social environment is changing, it is still true that software is not considered as important of a product as a publication, although projects such as the NSF funded URSSI project (http://urssi.us) seek to change this environment. Careers for such developers often lead into industry positions outside of the molecular sciences. While these careers can be very satisfying, it leads to some of the best computational engineers being enticed out of the very field that needs them.

Opportunities/possible solutions

Meeting these challenges can be daunting, however significant progress has been made throughout the molecular sciences community. The community itself has come to recognize that the traditional development methods have sustainability challenges and have sought out opportunities to change the environment. Indeed, the fact that NSF is now funding not only the MolSSI effort but other computational science software development efforts such as the Institute for Research and Innovation in Software for High Energy Physics (http://iris-hep.org/) and the Science Gateways Community Institute (https://sciencegateways.org/) is a significant sign that the environment is changing. These funding opportunities allow the community to implement significant changes in our software ecosystem that might otherwise not be possible. Here some of those possible solutions that we have been involved in as part of the MolSSI project will be outlined.

As one of the most significant parts of our agenda, MolSSI has engaged in a large education and outreach effort. These efforts include summer schools, educational workshops, and tutorials at the undergraduate, graduate, and postgraduate levels that focus on best practices in software engineering and applying those techniques to important topics in the molecular sciences. Several of our workshops also address programming issues related to changing hardware and high performance computing. To date, these activities have reached over 280 students and best software practices are becoming the *de facto* standard for the field. In addition, materials are made available online in a tutorial form to be accessible to those that cannot make the formal meetings or who have specific needs. In addition, MolSSI has developed an online resource page for best practices and partnered with the Better Scientific Software (https://bssw.io/) effort to provide general resources from the broader computational sciences community. In addition, MolSSI has awarded software fellowships to approximately 50 students with the specific idea of in-depth training in software best practices and engineering. These educational opportunities are making a foothold into the community and we expect there will be long term impact on how the community as a whole develops software as these students progress into the next phases of their careers.

MolSSI also sponsors approximately six software workshops per year reaching 525 participants to understand the challenges of software development in different pockets of the community. Most of these workshops are open discussions on bottlenecks to software development as well as issues associated with creating sustainable software. The workshop reports include specific,

actionable recommendations for MolSSI to aid the developments in the communities. Many of these recommendations - such as creating API standards and key infrastructure to help codes work together to enable world class science solutions - have huge potential to increase software sustainability in the community. In addition, these discussions continue to raise awareness of the challenges and possible solutions to software sustainability in the community. It should be noted that most of these workshops are lead by community leaders and not by MolSSI itself - MolSSI is a key partner in the workshops, but the leaders determine the primary topics of discussion.

MolSSI has also taken a leadership role in the development of code and data API standards. These common APIs will allow there to be a level playing field for all developers - enabling even the smallest software development effort to gain recognition for a unique feature or performance optimization. In addition, MolSSI is developing infrastructure software that uses these API to enable tasks that have proven to be difficult in our community - such as developing commonly accessible data sets using multiple quantum chemistry codes (QCArchive - https://qcarchive.molssi.org/) and for enabling the coupling of multiple quantum and molecular mechanics codes to perform very large, complex molecular simulations (MolSSI Driver Interface - https://molssi.github.io/MDI_Library/html/index.html). Ultimately, these types of standardizations enable fair competition with the users picking the most successful products for their needs.

Of course, this primarily MolSSI developed infrastructure software faces its own challenges of sustainability. However, MolSSI works to practice what it preaches: designing and developing with the users (mostly developers of simulation codes, but also actual end users); using thoughtful design to enable modularity, separation of concerns, reusability, and ease of use; use of standard APIs that enable a broad swath of developers to engage; documentation (both user and developer); use of distributed version-control systems, such as GitHub, automatic testing, and messaging tools to engage new developers and track issues; and building the user community through seminars, workshops, and personal communication. MolSSI will also continue to engage with commercial interests to provide training (as appropriate) and services associated with the software that is being developed. While the current model is a fully open source infrastructure, this may change depending on demand.

Finally, MolSSI is working to encourage new metrics within the field to reward those who take the software development path - other than a large salary at an industrial position not related to molecular sciences. These include efforts include encouraging developers to use DOIs on their software and datasets, raising the level of a software release to the same level as a publication, the use of data within tools like GitHub to show productivity, and encouraging the continued formation of positions related to scientific software development.