The Computational Molecular Sciences Software Development Community

Theresa L. Windus and T. Daniel Crawford

A community is not built in a single day or even in several years, but after many long years of effort and shared experiences. Even after a community has established itself, its members must tend to it faithfully for the community to continue to grow vibrantly. The molecular sciences community (MSC) is a case study in how communities can develop. The field consists of many different types of science including atomic and molecular electronic structure, biomolecular simulation, and materials. Each of these sub-fields has grown somewhat independently and have challenges that are specific to their field, but there are also shared concerns. While these fields have been reasonably well defined, they have not always enjoyed a sense of community, but of independent development and innovation. In fact, the rewards for individual effort have typically been much higher than those for collaborative efforts. Therefore, multiple programs developed with similar capabilities, but with specialization in different methodological areas or with different hardware. However, as each field has evolved, there has been a recognition that the molecular science community as a whole could advance much more rapidly if there were more cooperation in and between sub-fields. The molecular sciences community has developed its own software since the first computers became available, and these codes have evolved over time in conjunction with the hardware. Today, there exist many diverse programs ranging from sophisticated legacy software that have been developed over many decades to emerging code that uses modern software development techniques, but that may not be as fully functional.

A number of software developers in computational molecular sciences have worked for many years to form a community of developers that collaborate and move the science forward at an accelerated rate. Multiple workshops over perhaps 15 years have been held to encourage code and data sharing. However, only within the last 3 years or so has the larger community of developers begun to realize (as a community) that not only is collaboration useful and interesting, but that it is necessary for larger scientific impact. Why has it taken so long and what has finally motivated the community to decide that this is the right path? There are at least three factors that come into play: science, society, and security.

Science:

Until fairly recently much of the science that the MSC undertook was limited to relatively focused problems on a single time and length scale. Indeed, the solution of an individual scientific problem typically required a huge amount of new software. As the software capabilities have grown, the MSC has been able to tackle more advanced and difficult scientific questions. For example, instead of only being able to handle molecular reactions in the gas phase, now it is quite feasible to look at limited chemical reactions in the solution phase with very high accuracy as well. As another example, the biomolecular simulation capability has advanced to the point of being able to examine the dynamics of very large molecular systems for 100s of nanoseconds to millisecond time scales, but with less accurate methods for evaluating the interactions of the molecules and (generally) without molecular reactions (bonds being broken and made).

However, there are significant scientific drivers to perform reactive dynamics on very large systems with solvent or materials and with very high accuracy. One example is homogeneous or heterogeneous catalysis that might include microbes. These complex systems will have multiple phases with many different possible reaction paths, and a detailed understanding of the kinetics of these systems requires dynamics capabilities. The only way to simulate such systems accurately is to combine many different methods together, which requires expertise in many types of models and programs. The science drivers thus demand collaboration between different groups and between codes.

Society:

Even with the science drivers, there is no reason that a whole community of developers should be or will be involved in collaborating. So, often the solutions that developed early in the MSC were "one-off" solutions – an interaction between two software codes that accomplished the specific task at hand.

These solutions ranged from very loose coupling, where one code would write a file that could be used by another code in a separate run, to very tight coupling, where the two codes were compiled into a single executable to run the task. Often in this latter solution, one code would be the "driver" and the other one would be integrated into the driver code. Since the individual developers had a large stake in their own codes, these interfaces were developed to work best with the particular software and with the preferences of the developers involved. Due to the diversity of software and developers in the MSC, many of these one-off solutions developed redundantly – different combinations of codes to accomplish the same type of task. One example is the array of hybrid quantum mechanics/molecular mechanics (QM/MM) methods that combine accurate electronic structure methods for the reactive site with the less expensive MM methods to represent the rest of the chemical system. Many QM/MM one-off solutions have been developed, each with its own interfaces, data management approaches, and parallel computing methods. However, there are very few solutions that involve multiple QM codes or multiple MM codes. In part, this is due to the many interfaces that solve only the particular problem at hand.

This is where there has been a huge revolution in the community's thinking: instead of creating a solution that will only work for a few software codes, is there a solution that will work for many or all of them? Can the community define standards that will allow all QM codes to work with all MM codes? The advantage to such standards, for example, is that the community can then utilize a unique capability of a particular code or use a package that runs especially fast on the available hardware. Adherence to such standards would allow a new method available to a much larger community much more rapidly. In addition, the community as a whole could concentrate on the unique challenges that are presented by the interface and extend that interface to accomplish even more complex science. This is a bottom-up approach where the software development community has specific reasons to work together for mutual benefit.

Security:

The unfortunate reality is that economics often drives research and how it is carried out. One needs secure funding streams to support staff, postdocs, and graduate students and ultimately solve scientific problems. In the past, each investigator solicited his/her own grants to do his/her own research and develop his/her own software. Security was obtained by making your own software the best in the field and in competing with others for the funding to do so. However, in the MSC, it has become abundantly clear that we no longer have the luxury of developing everything individually from scratch. As much as one wants to create and innovate in all areas of the science, it is very difficult (and often impossible) except for the largest of research groups to secure the necessary funding to do so. As research support becomes more difficult to garner, and as funding agencies look to finance collaborative groups and ever-larger center proposals, it is imperative that the scientists and developers work together effectively as a team. Collaboration must be broad and effective to accomplish the scientific goals of the project. This doesn't negate the need for excellent individual research and software development, but it adds a new dimension to the research activities. Additionally, multiple agencies have provided funding to perform interface work in the sciences, notably the DOE SciDAC and NSF SI2 programs. Security is a top-down motivation that mandates that scientists work together.

Conclusion:

While there are still many issues to be dealt with in the MSC concerning standardization, interoperability, and program management, the MSC has become a true community with common goals and aspirations. This was accomplished through both a bottom-up approach with science and society being the main drivers and through a top-down approach from the funding security issues. Over many years of development, these three factors have converged in the MSC to accelerate new advances in science.