Lessons Learned Integrating Open Source Software in a Commercial Life Sciences Workflow Product

Scott Markel, Ph.D.
Principal Bioinformatics Architect
SciTegic
smarkel@scitegic.com

Bioinformatics is blessed with many open source tools that address the needs of our community. There are also *de facto* standards like NCBI's GenBank file format and BLAST program. And, of course, our customers expect access to all of these.

So what happens when a commercial life sciences workflow product makes heavy use of open source and publicly available programs?

Customers expect our product to be stable and professional quality, even when the third party tools we use aren't. There are times when we need to correct a bug in our version immediately and then change our fix if the third party tool is modified differently by the community. Our regression tests are helpful when upgrading an open source program. Due to the high throughput nature of workflow products, these applications are good at finding exceptional cases. We've been able to provide feedback and test cases for the tools we use, and, if we can, we propose a fix.

Our product has the requirement to run on both Windows and Linux, so there are the expected operating system issues. Many third party programs have a distinct Unix bias. This can make implementation on Windows a challenge. For example, the _program_list function from Bio::Factory::EMBOSS exits if the operating system is either Windows or Macintosh. Command line invocation of programs through Perl can also be a challenge. There are multiple ways to invoke programs from Perl: system(), backticks, pipes, Win32::Process::Create. All can have different behavior on Windows. Thankfully, Cygwin libraries can help by allowing Unix-like executables to run in a Windows environment.

The open source communities have been especially helpful during our development efforts. The open source advantages are well-known, including mailing lists that provide quick answers and software that has been tested. Of course, to be fair the latter usually means that common things have been tested by many people, while uncommon things may not have been tested much at all. Sometimes the community is very small, e.g., André Blavier and EMBOSSwin.

Other issues that must be dealt with include how to present command line options and error messages to users. Programs can have cryptic parameter values, e.g., genetic code numbers that need to be replaced with meaningful text. Some programs like Primer3 have large numbers of parameters. We split these into basic and advanced sets so that novice users aren't overwhelmed with detail. Our users expect error messages to be meaningful, allowing them to understand what went wrong and what to do about it. If a third party tool merely provides a stack trace, we may have to replace or augment it to make our software more user friendly.

Of course, every development effort has irritating little things to deal with. Examples of ours include the following.

- HMMER's hmmalign reformats NCBI's standard FASTA ID format. gi|460966|gb|AAA18225.1 becomes gi_460966_gb_AAA18225_1

- ctrl-A characters in NCBI's nonredundant database are illegal chars in any XML encoding.

- Swiss-Prot entry P03393 has the gene name "ENV". This wreaks havoc since the string gets used as a hash name.

- Fuzzy locations in GenBank entries. They're rarely used, but need to be correct when they arise.

We're strong believers in open source and publicly available tools and see the necessity of integrating them into a commercial product. Our community thrives on them, distinguishing us from other communities that rely more heavily on proprietary software, e.g., cheminformatics.