The Benefits of Open Source

Alan Westhagen
Software Architect
RFPK Software Team Leader

The System for Population Kinetics (SPK) is software being developed by the RFPK Laboratory, in the Bioengineering Department of the University of Washington, under grant *P41 EB-001975* from the National Institutes of Health (NIH) of the U.S. Department of Health and Human Services. This white paper presents the rationale for the choice by RFPK of an open source development and distribution model for this software.

Table of Contents

Introduction

With the traditional model of software distribution, source code was assumed to be private information. Software was distributed as binary object code, which would reveal its inner structure only through very difficult analysis of CPU instruction sequences. Users or third-party developers who wanted to modify the software were constrained from doing so by intellectual property laws, even when their efforts in reverse engineering were otherwise successful.

The open source distribution model reverses the old assumptions. Source code is not secret. A copy of the source code is always published as part of the distribution package. Anyone who has a copy of the code is authorized to redistribute it. Anyone is free to modify the code. If they do so, they are under no obligation to share their enhancements with anyone. If, however, they distribute a modified version of the original, they must include in the distribution the altered source code. Co-development is strongly encouraged.

Although this white paper will be primarily concerned with practical considerations, it is appropriate at the onset to acknowledge the equally important philosophical dimension. In a profound manner, open source mirrors the academic ideal of free exchange of ideas through publication and discourse, and the furthering of knowledge as a cooperative endeavor in ways that closed source does not.

Goals

In the interest of developing this rationale in a top-down manner, it is best to start with the purpose of the software itself.

SPK exists *to improve the quality and expand the application of population kinetic modeling in biomedical research.*

Important practical benefits will be a reduction in the cost of developing drugs, as well as improvements in their safety and effectiveness.

The Case for Free Beer

Open source software is often described as *free* software. What is meant is *free, as in freedom,* and not *free, as in free beer*. The latter definition is, nevertheless, highly relevant to the distribution of SPK. Some of the potential users of SPK have the resources to pay handsomely for critical software. This is true, one would suppose, of the pharmaceutical industry. Drug companies, however, will not buy software which has not already established its credibility by use in the broader biomedical research community. For this reason, RFPK has chosen a strategy of setting the purchase price of SPK to zero, in order to maximize the number of users.

It could be argued that by giving away its software RFPK is foregoing income which might be used to hire more programmers, to make the project self-sustaining, or even to provide a profit. There is no guarantee of success in the software business, however. Many firms have produced excellent software but have nevertheless failed for marketing reasons, and their software has disappeared along with them. RFPK believes that wide adoption of its software and its long-term viability can be achieved without subjecting the project to the vagaries of the market.

The Case Against Free Binaries

As an alternative to giving away source code, RFPK considered giving away object code instead. There are several reasons that this option was rejected:

- Open source greatly reduces the cost of developing and distributing complex software, such as SPK. (This advantage will be described in subsequent sections.) If RFPK were to exclude source code from its distribution, it would become ineligible for most of the benefits of open source development.

- In addition to a workstation and an Internet connection, an SPK user needs access to an SPK service provider. The service provider offers a web site, servers and a computational cluster. RFPK is itself a service provider. For the project to achieve the goal of wide adoption, there must be many more SPK service providers. A service provider might be embedded in an academic research group such as RFPK. It might be part of the IT department of a pharmaceutical company. It might be one of the offerings of a consulting or computer services firm.

  Providing a service such as SPK represents a considerable investment, especially if a powerful computational cluster is involved. Before making this commitment, a potential service provider would want some assurance that SPK would continue to evolve and to be maintained. RFPK is a first rate organization, with excellent prospects for obtaining a renewal of its current grant and for obtaining additional grants. Even so, some might see the small size of RFPK as a risk factor.

  With open source, all interested parties have access to source code. In the unlikely event that RFPK would no longer be able to continue the development of SPK, other organizations would be positioned to step in to fill the gap.

The Case for Open Source

Open Source provides RFPK with the freedom it needs to develop a complex product on a modest budget.

Freedom to Include the Work of Others

SPK includes many other pieces of software, some of which represent much more development effort than the code that RFPK itself has written. The incorporated modules include:

- *MySQL* relational database management system
- *Tomcat* web server
- *NetBeans* interactive development environment
- *GNU RCS* revision control system
- *Atlas* (Automatically Tuned Linear Algebra Software)
- *CppAd* c++ automatic differentiation library
- *GNU Operating System*

All of this software can be freely distributed as part of a system that is open source. There is no need to negotiate with the copyright holders over price or conditions. Inclusion of closed source software always requires negotiation preceded by cost and benefit analysis.

Consider for example the MySQL relational database management system. As part of an open source application, it can be freely distributed without the payment of any royalties. When part of a proprietary system, however, it can only be distributed under a commercial license, which requires the payment of fees similar to those charged for other database management systems. If SPK were not open source, it would be necessary to do a cost and benefit comparison between MySQL and similar systems, such as Oracle, DB2, SyBase and SQL-Server. It would then be necessary to negotiate prices for placing the software on development machines, test machines and, finally, on the customers' machines. The price of the database management system would be loaded on top of the price of SPK, making the total package more expensive for the customer to acquire. Having to deal with business issues such as these would inevitably draw resources away from development.

The SPK service is delivered over the world-wide-web. MySQL, Tomcat, and the GNU operating system are widely used in that environment. Although Microsoft products dominate on the desktop and in the office, open source products dominate the Internet. SPK concedes nothing in terms of quality or usability by including these excellent products.

Freedom to Share the Work with Others

With closed source, a software company is always extremely protective of its intellectual property. It is very rare that a company opens its source code to outsiders, and even then it is not to enable them to modify the code, but only to better understand how it works or how to interface to it. They usually expect to be paid for this privilege and to be indemnified in case their secrets are leaked to third parties.

With open source, the situation is reversed. Every recipient of the software has a right to the source code and the right to modify it. There is no need to negotiate price and conditions. There is no concern over the leaking of information that is already public. Anyone with the software is a potential co-developer.

The tools that are used to develop open source products are themselves open source. The cost of obtaining these tools is low or non-existent. Anyone with a Linux computer, for example, already has most of these tools installed on her machine or can easily download them from the Internet. This is another way in which the threshold for becoming a co-developer is very low.

Freedom from Proprietary Lock-In

The strategy to make it difficult or impossible for consumers of computer technology to change system vendor is known as proprietary lock-in, At one time, this strategy was pursued successfully by all major computer manufacturers. Software developers were enlisted, whether willingly or unwillingly, in this endeavor via the means of the *application programming interface (API).* To develop software to run on the systems of a particular computer company, a developer had to use the API of that vendor. The software would run on the systems of that company and none other. This strategy partitioned the world of software development along API lines. Because of the intentional incompatibility of the APIs, most software houses were forced to develop software for the platform of a single vendor. It would then be impossible for consumers who purchased the software to switch computer manufacturers. They were locked in and could no longer benefit from competition between vendors.

The Unix operating system was revolutionary, not because it was more powerful, or more efficient, or had better features, but because it was *portable*. Within a few years of its creation at AT&T Bell Laboratories, it had been modified to run on most of the computer processors available at the time. Software development needed no longer to be *balkanized.* Systems written to the Unix API could be easily ported to any Unix system, whether the hardware came from IBM, Digital, Hewlett Packard, Sun, SGI, etc. Consumers loved it, because they now had the option of changing hardware vendors. The old-line computer manufacturers hated it, and did everything in their power to impede the process of migration to Unix.

By the mid 1990's, the battle between Unix and the old proprietary operating systems was over, with Unix the victor. Those computer companies which had first opposed Unix and then had impeded its adoption were now solidly committed to delivering Unix on their hardware and developing applications using the Unix API. At the same time, these companies underwent a transformation from being primarily manufacturers to also being systems integrators and consultants. In this new environment, an operating system which could run on any hardware made integration of systems much easier.

At the same time that Unix was consolidating its dominance for the large and medium sized computers used as Internet servers, database servers, and enterprise application servers, another force was at work. Microsoft recognized that desktop computing and office automation would be the major growth area for information technologies. By developing Windows as an operating system that could provide an easy-to-use graphical user interface for the desktop and a platform for office automation, Microsoft asserted its dominance at the low end of the computer spectrum.

SPK is a complex system, made of a number of components. These can be divided between the *server side*, which runs on server machines and high performance computing (HPC) clusters, and the *client side*, which runs on the end-user's workstation or laptop.

For the server-side, there are today for all practical purposes, only two APIs for software developers to choose from. The Unix API is supported by all Unix and Unix-compatible systems, including Linux and Apple Macintosh OS-10. The Windows API is supported by the Microsoft Windows operating system. The two are incompatible.

The choice of server-side API determines the processor architectures on which the software will run. Programs written to the Unix API can be installed on nearly every processor architecture in existence, including Intel Pentium and Itanium, but also Sparc, Alpha, PowerPC and IBM Mainframes. Software written to the Windows API runs only on Pentium and Itanium.

The choice of API also determines the operating system that must be used. For SPK, this turns out to be of prime importance, because almost all HPC clusters run Unix or Unix-compatible operating systems rather than Windows. SPK is developed to the open source GNU version of the Unix API. This is the most portable of all APIs. This choice enables SPK to run on practically all clusters available.

For the client side, many end-users of SPK prefer to use Windows as the operating system for their workstations and laptops. This is a choice that RFPK respects. Proprietary lock-in is avoided through the use of Java, the *write once, run anywhere* technology, which allows the client side of SPK to run not only on top of Windows, but also on Apple Macintosh OS-10, Linux or Unix, depending on the user's preference.

The Impact of Open Source

Up to this point, this paper has focused primarily on how the decision to develop and distribute SPK as open source affects RFPK itself. This decision also affects other constituencies, including end users, service providers, consultants and software houses.

End Users

The impact on end users is positive and unambiguous. They obtain the client software for free and the service at a price that should be competitive because of the low threshold of entry for service providers. In fact, computer-savvy end users can obtain the entire SPK package for free and install it on their own hardware if that is preferable to using a service provider.

Service Providers

Service providers can download SPK for free and install it on their hardware. Because RFPK has chosen to develop SPK using open source GNU tools, it is possible to install the software on just about any hardware that the service provider has available. The requirement that the operating system be either Unix or a Unix-compatible system such as Linux or Mac-OS10 is not a barrier for most service providers, who already have in-house Unix expertise. As discussed above, the the fact that open source distribution creates a community of individuals and organizations that would continue the development of maintenance should RFPK go out of existence, eliminates that risk to service providers.

Consultants

SPK can be a real boon to consultants, especially if they choose to become service providers. Because SPK is a web based service, it is easy to differentiate a particular SPK from others through specialization of the web site and through the provision of proprietary PK models. It is important to note that models are not part of SPK. They are, rather, the intellectual property of the person or organization which develops them.

Software Houses

There remains opportunity for developers of proprietary software to add value to SPK by replacing entire modules. Architecturally, SPK can be viewed as a cooperating group of loosely-coupled independent processes. The interfaces between these processes are well documented. A software house could, for example, replace the SPK client program with its own proprietary client, which might feature a different type of graphical user interface or which might integrate seamlessly with its other products. A software house that specialized in getting the most out of computational clusters might develop a proprietary replacement for the SPK run-time daemon which might do a superior job of optimizing work flow.

Summary

In conclusion, open source is the right choice for RFPK, because it significantly enhances its ability to achieve its goal, given its limited resources. Other groups benefit as well, because open source creates a community to which all constituencies can contribute and from which all can draw benefits.