Repositioning SPK as Open Source Software

Abstract

The greatest challenge faced by the SPK software product is a scarcity of users. This paper proposes a reorientation which should provide many benefits, including the following goals:

- Many new users.
- Cooperative development of SPK, enlisting collaborators, partners and users.
- Harnessing of massive amounts of cheap computer resources to optimize population kinetics models.
- Ensuring a central role for RFPK in the future of SPK, as long as RFPK continues to exist.
- Ensuring the future of SPK, even if RFPK ceases to exist due to non-renewal of its public funding.

Table of Contents

The Current Distribution Strategy is Not Working **1**
The Free Object Code Alternative **2**
Source Code Alternatives **2**
Benefits of Open Source **4**
The Process of Transformation **6**
Future Directions **7**
Summary **8**

The Current Distribution Strategy is Not Working

The initial beta-test version of SPK was not released to end users. Instead, copies were provided to seven for-profit consulting and software firms working in the field of biomedical research. The rationale for limiting distribution to commercial firms was that, as a publicly funded research facility, RFPK ought not compete with the private sector. Instead, RFPK would form partnerships with the firms which otherwise might have been competitors and would reach the end users through them. These firms, which will be referred to as *partners* in the rest of this paper, would add value to SPK, in terms of models, user interfaces and consulting.

This distribution strategy has not worked well thus far. SPK has been in the hands of the partners for more than six months. In that time, none of the firms has made the effort to run the software. There are a number of possible reasons for this:

- *Level of effort to create a model.* Although several canned models were included in the release package, to seriously evaluate SPK in terms of its own needs a partner would need to create one or more models of its own. This is not a simple process; it requires considerable effort and ability at programming in the C++ language.

- *Uncertainty about the future of SPK* There are fewer than six months remaining on the National Institutes of Health (NIH) grant that funds RFPK. Even if this grant is extended for an additional five years, as RFPK has requested, the disposition of the software after that time is not clear.
- *Uncertainty about licensing.* The current license only allows the partner firms to evaluate and test SPK for a fixed period of time. Before investing any resources, most firms would like to know what the eventual licensing terms will be. In fact, given the uncertain future of RFPK, most firms would like source code, at least in case SPK were no longer being enhanced and maintained to their satisfaction.
- *Inappropriate platform*. One partner who has provided feedback indicates that its clients would be interested in running the software on Linux clusters. Population models can take many days to run on Windows workstations. There is a clear need to provide a more powerful platform.

The net result of the decision to distribute the software exclusively through partners is that SPK has no users at all, outside of RFPK. It will not have users until the partners invest resources in evaluating and adapting the product to their business needs, or until the distribution strategy is redefined.

The Free Object Code Alternative

Free distribution of object code has been suggested as an alternative distribution strategy. Free binaries could be offered to the partners alone, or to any organization requesting them. In the case of the partners, being freed from any obligation to pay a license fee might be seen as favorable to increased sales and profits. In the case of other researchers, this mode of distribution would put SPK in their hands more quickly and cheaply than if the software had to first pass through the hands of partners. In any case, it would clear up some of the uncertainty about licensing.

Although this strategy could provide some benefits, it still would suffer from lack of any guarantee about the future enhancement and maintenance of SPK, because the development of the product would remain the exclusive domain of RFPK.

Source Code Alternatives

The best way to neutralize fears about the ability of RFPK to enhance and maintain SPK in the future would be to offer source code. This way, a partner or an end user organization which felt the software was not being enhanced or maintained in response to its needs could undertake this work itself. If care were not taken, however, in specifying the terms under which the source code would be licensed, additional problems would be created.

There are two principal ways to license source code:

1. *Not open source.* With this alternative, there is no requirement for those who make changes to the source code to make these changes available to others.
2. *Open source.* An open source license requires that those who distribute the software to others must provide the source code, including any modifications they have made.

Source Code Distribution Without Open Source License

Source code for software products has been distributed along with object code for many years. For large expensive commercial software systems, this is the norm. Cus-

tomers require access to source as a guarantee that their investment will not be lost if, for whatever reason, the vendor can no longer maintain and enhance the software to agreed upon standards. For their part, however, vendors are very reluctant to allow anyone other than themselves access to the source because they fear their intellectual property and their proprietary ideas will fall into the hands of competitors. The procedure that is often followed in the commercial software marketplace is for a copy of the source to be placed in *escrow* with a third party, to be released only in case the vendor goes out of business or fails to support the product as agreed upon.

A major disadvantage of licensing SPK with a source code escrow would be that it could provide an incentive for a partner to take the software owner, namely the University of Washington, to court if the partner were dissatisfied with the performance of RFPK. This could easily occur, given that the aims of RFPK and those of most for-profit firms do not coincide. If the unhappy partner won its case, it could wrest control of SPK from RFPK.

Providing source code directly, without escrow, to partners or end users would be even more disastrous. Within a short period of time there could be several groups developing their own versions of SPK, all incompatible with each other and with the original. SPK would give birth to several proprietary products, and RFPK's software efforts would thus be marginalized.

Source Distribution With Open Source License

An open source license would protect partners, users, and RFPK. Many successful open source projects have amply demonstrated this. A discussion of some of these projects and the many advantages of software development under open source will be summarized in the next section. For the time being, it should be sufficient to point out that the open source development model is superior to the one currently being followed by RFPK.

The basic principal of open source is very simple. *If you distribute software that you received under an open source license, you must provide the recipients of the software with the same rights you were accorded.* In other words, if you distribute the software, whether in its original form or as modified, you must not only pass along to the recipient the source code, but also the right to distribute it.

The open source license would solve two problems immediately:

1. Any incentive a partner or any other user might have to "kidnap" SPK in order to turn it into a proprietary product would vanish.
2. Doubts about the ability of RFPK to continue the development and maintenance of SPK would no longer have critical importance, since any of the partners or users or any collection of them could take over this responsibility.

Would not SPK break up into a number of incompatible versions? No, because any group that enhanced SPK would have a strong incentive to submit the changes to RFPK for incorporation in a future release, in order to avoid having to maintain the software on its own. RFPK would remain the focus of SPK development. It would continue its own program of enhancement, but added to this would be the integration of enhancements by others.

Why would partners have any incentive to make enhancements, if they were required to give away their source code to competitors or potential competitors? Because the open source license could be written to insure that models and user interfaces were exempted from the requirement to distribute source code. There is already a strong precedent for this. The most famous open source license, the *Gnu General Public License (GPL)* which covers an enormous body of software including most of Linux, is one of the most strict open source licenses, in that it requires any derivative work to be distributed as open source. The *Gnu Library License (LGPL)*, on the other hand, allows

works which are linked to the Gnu function libraries to be distributed as proprietary binaries. RFPK could pattern its license to be similar to the LGPL.

Would not RFPK and the University of Washington forego a lot of revenue by distributing SPK for free? Not necessarily. Many of the largest commercial software enterprises in the world, including IBM, Oracle, Sun and Hewlett Packard are actively supporting, distributing and building products around open source software. The key for RFPK is to get users; then biomedical consulting and software development work will follow. For instance, a partner may require a certain enhancement to SPK in order to satisfy the requirements of its clients, but the enhancement might be low on RFPK's priority list. The partner could develop the enhancement itself. It might make more sense, however, for the partner to provide funds so that RFPK could hire additional developers to produce the enhancement, while continuing to proceed with its original list of priorities. This sort of thing happens all the time in the open source world. Many millions of dollars of open source development resource are being provide by commercial enterprises.

Benefits of Open Source

1. Users are absolutely essential to any software development project. Offering SPK for free via the RFPK web site would give the quest for new users a tremendous boost. The need for these users to visit the web site often, for updates, news, and technical information related to SPK would greatly increase the visibility of RFPK in all its aspects.
2. Experience with other open source products, including the Linux operating system, the Apache web server, the MySQL and PostgreSQL database systems, and the Sendmail electronic mail system has shown that in spite of the fact that the software is given away free, much revenue can be brought in from related services.
3. Open source development methods are equal to or superior to traditional commercial development methods, producing results in less time, at lower cost, with higher quality and excellent usability. These methods have several powerful advantages over traditional methods:
    a. Open source development teams differ radically from their traditional counterparts. Rather than requiring all team members to work in close proximity to optimize spoken communication and managerial control, open source teams are often constituted of members who live in different cities and continents but communicate via the Internet. Development tools have evolved to support this new style of distributed collaboration. In the University of Washington setting, these new assumptions would coincide with the goals of flexible time and telecommuting. In the RFPK context they would open the door to cooperative development between the core team and contributors around the world.

       To gain a bit of perspective on this, consider SourceSafe, one of the Microsoft development tools currently in use by the RFPK software team. Four team members are located in the same office, while one member is in another building. Although the two buildings are connected via the University of Washington network, it is a common occurrence for the in-office developers to be locked out of the development system while their colleague is accessing it. On a number of occasions, this situation has persisted for hours. The problem would only be worse if a need arose for one or more team members to work from home or from other locations. The basic flaw is that the Microsoft tools were designed to sup-

port traditional centralized development teams rather than distributed collaborations.

In passing, it is worthwhile mentioning that open source software development tools are themselves open source software and, as such, are free of charge.

b. Users are essential to the testing and debugging of software. At present, RFPK is at an enormous disadvantage, having virtually no users at all. Free distribution of the software over the Internet would give us the best opportunity to rectify this deficiency. These users would be self-selected and, as such, would have a professional need or interest in the software. These are the people who would be most likely to use the software and to help shape it to their own needs.

c. Support of a user base takes resources. RFPK is small and is not likely to grow without additional revenue. Certainly some users (drug companies, for example) will be well-funded and able to pay RFPK or one of the commercial partners for support and consulting. Others however, will have to be self-supporting. This will be completely feasible, given that they will have access to the source code and to shared information on the RFPK web site.

4. Linux is the preferred open source development platform. If the decision were made to offer SPK as an open source product, it would be logical to port the software to Linux. This move would have many beneficial side-effects:

   a. Linux runs very efficiently on ordinary Intel-based PC hardware. It also runs on more powerful processors, including Alpha, UltraSparc, PPC and IBM mainframes. In terms of upward scalability, Linux far exceeds any Microsoft operating system. This is highly relevant to SPK, since some population models will undoubtedly require large amounts of processor power.

   b. Unlike Microsoft operating systems, Linux is hardware independent. It currently is available on more than a dozen processor architectures, whereas Microsoft only supports the Intel X86 series. When a new, powerful, processor arrives on the scene, Linux is rapidly adapted to it. Unless the processor fits into Microsoft's business strategy, which is essentially the support of office desktops and departmental servers, it is unlikely that a version of Windows will be offered.

   c. The need for powerful hardware on which to run population models goes beyond the choice of processor. When asked whether or not it had evaluated SPK, one of the partners replied in the negative but did provide some valuable feedback. The partner said its own clients would be interested in a Linux version of SPK because then they could run population models on Beowulf clusters.

      Beowulf was a project started by NASA in 1994. The idea was to investigate the possibility of building super computers from cheap, generic computer components. The technology, which became known as *commodity off-the-shelf* (COTS), leverages mass-produced computer hardware, the Linux operating system, the Gnu software development tools, and various open source drivers and libraries.

      The approach has been enormously successful. By 2002, 49 Beowulf clusters were included in the list of the most powerful 500 super-computers. Beowulf clusters are widely used by universities and government laboratories. They have also found a home in private industry. Hollywood uses them extensively for digital film production. For instance, the ray tracing algorithms required for creating the digital images in the film Titanic were run on a Beowulf cluster of Alpha processors. Conoco uses a

large Beowulf cluster to analyze seismic data for oil and gas exploration. Locus Discovery, a bio-pharmaceutical firm, uses a cluster of 620 nodes, each containing two Intel PIII processors, to help design new drugs.

At present, RFPK owns a small Beowulf cluster, but has not made much use of it, given that there is currently no Linux version of SPK. This would change if RFPK adopted an open source strategy. The cluster would then become an invaluable development resource.

The Process of Transformation

The following are the major tasks that would transform SPK into an open source product. Many of these tasks would be performed in parallel.

1. *Convert software team workstations to Linux* and install the full complement of open source development tools and open source office tools.
2. *Accustom the software team to the open source development environment.* This would not be formal training. Facility with GNU development software and the CVS source control system and the Unix/Linux environment would accrue during the porting of SPK to Linux.
3. *Establish a Linux server* to provide a repository for SPK source code and documentation. This machine would not have to be new or powerful; any surplus Pentium class PC would do.
4. *Transfer SPK source code to CVS*. Every software development project needs a source control system, to enable several programmers to work on the same source files at the same time without wiping out each other's work, and to keep a complete history of all changes. Currently, RFPK uses the Microsoft SourceSafe source control system. The inadequacy of SourceSafe when used by developers in separate locations was described, above. The open source CVS source control system, on the other hand, serves geographically distributed development teams very well. Version 1.0 of SPK would be entered into CVS which would control the source from that point onward. No attempt would be made to transfer the history of the development of previous versions from SourceSafe to CVS. Instead, the history leading up to Version 1.0 would be maintained as long as necessary under SourceSafe on the current Microsoft server.
5. SPK currently is built around the NAG optimizer, which is proprietary code. Before SPK can be distributed as open source software, a new optimizer must be developed. This is an opportunity to leverage the considerable expertise of the RFPK Mathematics Team in this area.
6. *Port SPK to Linux.* SPK is written in C++, which is a language which originated in the Unix environment. C++ applications interface with the operating system via a library of Unix system calls. When C++ runs in a non-Unix environment such as Microsoft Windows or NT, a library is provided which emulates Unix system calls. In effect, SPK is already designed to run under Unix.

   As an open source rewrite of Unix, Linux conforms to nearly all Unix standards and conventions, SPK will compile and run under Linux with very little change. There are a few Microsoft-specific areas in the code, such as file-locking interprocess communication for the parallel implementation, and the Matlab interface, but these are minor and readily identifiable.

7. *Establish an ftp server*, or acquire space on an existing one, from which users would download free copies of SPK, related software, and documentation. This machine would not have to be new or powerful; any surplus Pentium class PC would do.
8. *Upgrade the RFPK web site* with a section dedicated to the dissemination of SPK software, documentation, and collaborative communications.

Future Directions

The reorientation of the RFPK software efforts towards open source, Linux and Beowulf will open up some very promising directions.

Emphasize Scalability

Population kinetic models can vary enormously in the amount of processor resource required to run them. The low end of the scale presents no difficulty. The problem can be run on a PC using convenient but inefficient languages such as Matlab, Octave, S and R. The high end presents a major challenge but SPK is already positioned to make a major contribution.

SPK can be thought of as an engine for running a bio-kinetic models in order to calculate certain unknown model parameters. Models are software separate from SPK, although they must be designed according to certain specifications in order for SPK to exercise them.

SPK is written in C++, a programming language which is capable of providing efficiency in numerical calculation that equals or surpasses Fortran, the language traditionally favored for this type of work. As time goes by, improvements in the efficiency of SPK will continue to be an important priority for RFPK.

Complex models with lots of data require lots of processor resource. For maximum efficiency, such models should be programmed in C++. Even so, a model may require weeks of continuous running on the fastest of PCs. It is obvious that such long running times are unacceptable to many researchers.

Just as with many other intellectual and creative endeavors, the largest cost in kinetic modeling is the cost of human talent. A close second is the opportunity cost incurred when research and pharmaceutical products are delayed. The cost of computer resources is at best of tertiary importance.

A highly scalable version of SPK would adapt automatically to the computer resources available, whether running on a Linux workstation or on Beowulf cluster. On a cluster, the divisor by which the run time of a problem would be shortened would approach the number of nodes employed. A user organization that required faster turn-around would simply purchase additional nodes for its cluster. At present, nodes cost less than $1000 apiece. The cost effectiveness of this strategy will double at least every 18 months for the next two decades, according to Moore's Law.

Emphasize Modeling Languages

If the goal is enhanced productivity for researchers, we must support modeling languages which are more expressive of mathematical and scientific concepts than is C++. Among these languages are the following:

- Matlab, a general mathematically oriented language, widely used by scientists, engineers and mathematicians.
- Octave, an open source language which shares much of the Matlab syntax.

- S and Splus, widely used for statistical analysis and modeling.
- R, an open source version of S.
- SAAM II, a graphical modeling environment for individual kinetic modeling.

These languages are nowhere near as efficient of computer time as C++, but they are surely more efficient of researcher time. In many cases, adding enough additional nodes to a Beowulf cluster to make up for the lowered computational efficiency would be a very sound investment.

SPK already has a Matlab interface. A migration to open source should make Matlab a viable alternative to C++ on Beowulf clusters. Other languages can be added in response to user demand.

Develop an SPK Server and API

RFPK should develop software for the efficient and convenient management of a Beowulf cluster as an SPK server. Models would be sent to the server via the Internet and queued for execution. If sufficient nodes were available, the model would run immediately; otherwise it would wait its turn for access to nodes. Status and results would be available over the Internet.

Access to these services would be via a network-oriented *application programming interface* (API). RFPK would develop utilities to access these services via the API from PCs, MACs, Unix/Linux workstations and the world wide web. Equally important, however, would be the availability of this API to partners and collaborators for the modeling tools they would develop.

Summary

The goals listed in the abstract are all achievable, as well as many other side-benefits, if SPK is repositioned as an open source project. This move should be undertaken as soon as SPK version 1.0 is released, which would be the culmination of the long and intense development effort under the first NIH grant. Version 2.0 should be open source, Beowulf capable, and domain-specific language friendly.