A Major Course Correction

An analysis of the disappointing response to the beta release of SPK has lead to major changes in the system architecture, the mode of distribution, and the methods of software development. With the goals of increased *usability* and *accessibility* at the forefront, SPK is being transformed into an Internet application, featuring automatic generation of computer-ready models from specifications that users provide via a scientist-friendly GUI, parallel computation on a Beowulf cluster, Open Source licensing, and free distribution.

Table of Contents

Lessons Learned from the Beta Release

During the 3rd quarter of 2002, the first beta-test release of SPK was distributed to six commercial partner firms, which were chosen for their prominence in the field of biokinetic software and consulting and for their interest in the project. The expectation was that these firms would evaluate SPK by developing test models typical of their own work and by running these models on the SPK platform. This was to be the first time that SPK was exercised outside of the University of Washington Bioengineering Department.

The results of the beta-test cycle were extremely discouraging. None of the partner firms was willing to expend the effort required to do an effective evaluation. The feedback that RFPK received was that SPK is difficult to use, that the documentation needs a lot of improvement and that, in the opinion of one commercial partner, a Linux version would have been more interesting than the Microsoft Windows version supplied, because then it might have been possible to run it on a powerful Beowulf cluster.

An analysis of this disappointing experience highlighted a number of possible problems:

1. SPK is quite difficult to use. It was designed to serve the needs of a particular class of user: namely scientists who themselves are skilled C++ programmers, or scientists who have professional C++ programmers at their disposal. Such users seem to be rare indeed.
2. It had been assumed that SPK would be developed into commercial products by the partner firms. The beta software was distributed to them without a clear understanding of what they would have to pay in terms of licensing fees. There was no guarantee that one of their competitors might not obtain an exclusive license. The future of RFPK itself was uncertain, which naturally raised questions about development and maintenance of the software should funding dry

up. In retrospect, it is not surprising that none of the selected firms saw investing in the evaluation of SPK to be in its interest, given these uncertainties.

3. As a Microsoft Windows application, SPK was not capable of being run on more powerful systems, such as Beowulf clusters, Unix multiprocessors and IBM mainframes.
4. The fact that none of the six recipients of the beta release saw fit to evaluate SPK does not prove that there are no potential users in the scientific community. Results might have been different if SPK had been distributed to a much wider audience.

New Goals: Increased Usability and Availability

Out of this analysis came the realization that the most important changes that RFPK could make would be to dramatically increase the usability and availability of SPK to the scientific community. We decided that a multi-faceted approach would be the best means of achieving this:

1. Make SPK much easier to use. C++ expertise should not be required. Instead, the scientist should be able to use the language and modeling conventions that she prefers, whether Nonmem, Matlab, Octave, SAAM-II, S-Plus or R.
2. Remove all doubt about the licensing and future support of SPK, while preserving the opportunity for commercial firms to earn a profit by adding value.
3. Modify the SPK computational engine so that it can efficiently utilize Beowulf clusters, Unix and Linux symmetric multiprocessors, and mainframes. Develop client software to allow scientists to specify models and to view and analyze results on whatever workstations they prefer, including Windows, Macintosh and Linux.
4. Make it possible for the largest possible group of researchers to reap the benefits of SPK by distributing the software free via the Internet.

A Three-Tiered Architecture

The new goals could, of course, be achieved in a number of ways. The course that we chose is based on a desire to harness the technology of the World Wide Web. It is an understatement to point out that web technology has revolutionized the manner in which nearly everyone interacts with computers. Given the limited financial resources of RFPK, it is especially fortuitous that much of the web runs on free software. RFPK can derive enormous leverage by utilizing free technology to transform SPK to a *three-tiered architecture* similar to that used by most prominent web sites.

The SPK three-tiered architecture will consist of the following strata:

- Client tier: the *model design agent*
- Middle tier: the *application server* for population kinetics
- Foundation tier: the *computational server* for population kinetics

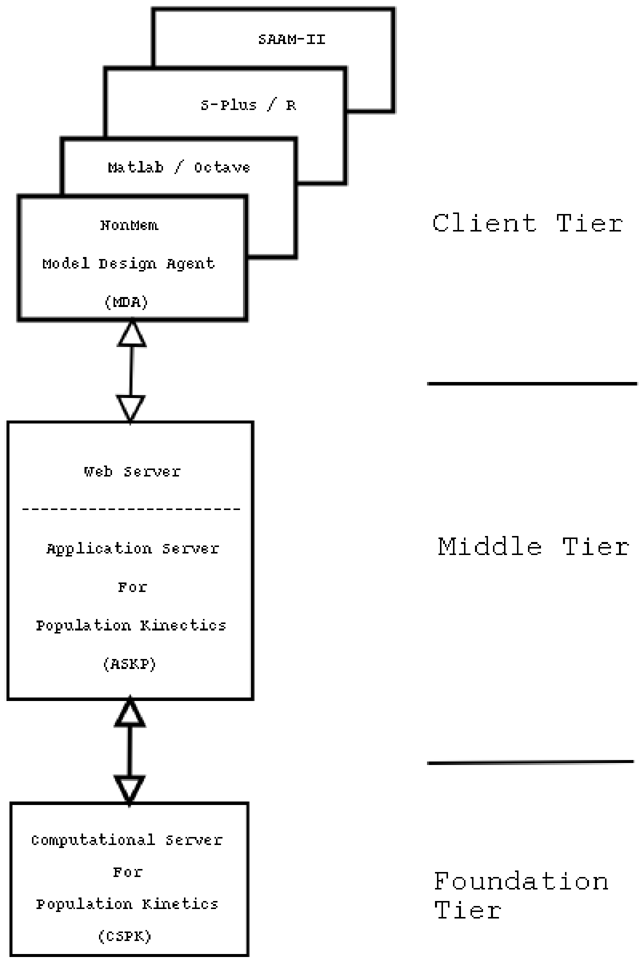

Figure 1. Three Tiered Architecture

Model Design Agent (MDA)

At present, SPK is a C++ class library. By itself, it is not a computer program and can solve no problems. It lacks a model and a model driver. The scientist is expected to supply these elements in the form of C++ source code. Experience has shown this requirement to be unrealistic. In the enhanced architecture, the model and driver will be generated automatically, to the scientist's specification.

Scientists who build bio-kinetic models use a number of different languages and tools. Principal among these are Nonmem, Matlab, Octave, S-Plus, R, and SAAM-II. These users should not be required to learn a completely new language in order to use SPK. Requiring expertise in C++ was not only unrealistic because of the difficulty of that language; it was also unrealistic because C++ is a language not normally used in bio-medical research.

The specification for a kinetic model requires the definition of expressions and data. The expressions represent mathematical functions such as the function for the mean, its derivative, or differential equations. Data includes measurement data, parameters, and other descriptive information. Each of the languages used to build or specify models has its unique syntax and characteristics. For instance, Nonmem uses a Fortran-like syntax for expressions, providing these and other required information in Nonmen-specific *control files*. SAAM-II has its *study* and *data* files. Matlab has its own expression syntax, and so forth. SPK will allow users to specify models using the conventions with which they are familiar.

As is true of most non-scientists, we can assume that the majority of scientists prefer to use Microsoft Windows to operate their workstations and laptops. A significant minority, however, prefer the Macintosh, Unix or Linux. Some use personal digital assistants. In the not-so-distant future, many will be running applications on their wireless phones. In order to serve the largest possible group of researchers, the enhanced architecture will eventually support all of these operating environments.

SPK will include Model Design Agents (MDA), which are easily installed programs with graphical user interfaces (GUI) that assist scientists to specify population kinetic models. MDAs will be available for each of the supported modeling environments, including Nonmen, Matlab, Octave, S-Plus, R, and SAAM-II. MDAs will be written using portable GUI development tools so that they will run with little or no modification on many operating systems, including Microsoft, Macintosh, Unix and Linux.

MDAs form the *client-tier* of the *three-tier architecture*. They will communicate with the middle-tier via the Internet.

Application Server for Population Kinetics (ASPK)

On her workstation or laptop a scientist will design her model. She will then transmit the model to a web server using the same technology employed by her web browser. This mode of Internet communication has a number of advantages, especially the ability to pass through any firewall that permits web communication. Running on the web server will be the *Application Server for Population Kinetics (ASPK)*, which is the middle-tier of the *three-tier architecture*.

Coupled with the web server on which it runs, ASPK will be the heart of the enhanced *System for Population Kinetics (SPK)*. Each SPK installation, whether the one under the auspices of RFPK itself or those that we anticipate being established by other universities, research organizations, and pharmaceutical companies, will have a *middle-tier* which will offer the following services:

1. *Host a home page for the particular SPK installation.* This page will include links to news and documentation about SPK. Those who administer this page will certainly want to include information about bio-medical research at their own institutions and elsewhere.
2. *Provide a personal portal to SPK services.* The scientist will log into her portal with a username and password. Once past the portal she will arrive at a personal web page where she will be able to do the following:
   a. Select an MDA from the list of those available: Nonmen, Matlab, Octave, S-Plus, R, SAAM-II, etc. After she makes her selection, a MDA window will appear on her screen.
   b. Within the MDA window, she will design models. When her model is ready to be run, she will have the MDA transmit the model specification to the ASPK. There it will be compiled into an executable model. If the model has compilation errors, an error message will be returned to the MDA to be reported to her. If the compilation is successful, the model will be queued for execution on the *Computation Server for Population Kinetics (CSPK),* which is described below.

c. Once a model has been queued for execution on the CSPK, the MDA is released so that the scientist can develop additional models if she wants.

d. When a model completes, the availability of results will be announced on the scientist's personal web page, along with a link for retrieving them. When she clicks on the link, an MDA window will be opened on her screen. The MDA will assist the scientist in converting the results into forms required by the analysis or visualization software of her choice.

e. Note that the portal architecture allows the scientist to submit a model from one computer and review the results from another. This will be especially useful to scientists who are on the move, but have models that take a long time to run.

f. The scientist will be able to store compiled models on the ASPK in her personal model library. The next time she wants to run this model, it will not be necessary to recompile it unless she decides to make changes. She will use the MDA to submit new data and parameters.

3. *Provide a site-wide library of models.* As explained above, individual users will be able to store functioning models on their personal home pages. They will also be encouraged to submit generally useful models to the site-wide model library. This will be a two-phase process. A model will first be submitted for review by the site administration. It will be analyzed for generality and reliability. If it passes, it will be documented and placed in the site-wide library for others to use.

Computational Server for Population Kinetics (CSPK)

The *foundation-tier* of the *three-tier architecture* is the *Computational Server for Population Kinetics (CSPK).* Basically, this is a server running the SPK library, to which models designed in the *client tier* and compiled in the *middle-tier* have been linked. It can be a single Unix or Linux machine, or a cluster of such machines. Each machine can, in turn have a single processor or many processors.

The first beta release of SPK included the capability of running a model distributed across multiple Microsoft workstations operating in parallel. This was possible because of the fact that the population kinetics problem lends itself to a simple and efficient factorization for parallel execution. The recognition of this fact is certainly one of the most significant achievements of SPK to date.

Running SPK on a group of workstations working in parallel has shown itself to be impractical for several reasons. In the first place, these workstations were purchased for other work, but SPK consumes nearly all available processor and memory resources when it runs, thus making it impossible for other work to coexist with it. In addition, managing the synchronization between machines of different capabilities and different usage patterns is complex and inefficient. The exercise did, however, demonstrate the concept. The lessons learned will be incorporated in the next iteration of the design, which will be targeted at clusters of similar Linux processors, under centralized control, and dedicated to computation.

The CSPK will incorporate software for managing a prioritized queue of models awaiting execution. It will be able to efficiently distribute work over a group of processors. It will also be able to work with just a subset of the processors in a Beowulf cluster, leaving the rest for other work.

Free Software

SPK, enhanced with the three-tiered architecture, will be offered free, in order to reach the largest possible community of scientists. Although it might seem that giving something away for free is a simple matter without nuance, there are a number of very important issues to consider. Here are some questions that we have asked ourselves:

1. Should we distribute binary copies of the system, or should we offer source code as well?
2. If we provide only binaries, what assurances can we give users that the software will survive RFPK in case the group's funding should disappear? What if users become dissatisfied with the level of competency or the commitment to ongoing development and maintenance of the product by RFPK?
3. If we provide source code, other groups can undertake or participate in the ongoing development and maintenance of SPK. If this occurs, how can we preserve a central role for the RFPK group? How do we minimize the tendency for multiple and possibly incompatible versions of SPK to spring up?
4. How do we preserve the opportunity for commercial software firms and research consultants to profit from enhancements to SPK and from the development of proprietary models, while keeping SPK itself from becoming a proprietary product?

Open Source

A distribution strategy that answers all of these questions is the now familiar open source software distribution model. This method of distribution is used for thousands of software products, including some of the best known: the Linux operating system, the *Apache* web server, the *Gnu* software development tools, the *Mozilla* web browser, and the *Mysql* and *PostgreSQL* database management systems.

Open source software is software distributed under a license that says, essentially, the following:

- Source code is available free-of-charge or at most for the cost of making copies and shipping them.
- The recipient is free to enhance or modify the software as he sees fit.
- The recipient is free to redistribute the software under the same license as it was received. In other words, the next recipient gets source code, and is free to modify and free to redistribute it.

Because enhancements must be shared upon redistribution, an open source license protects against the creation of proprietary versions. It encourages cooperation on enhancements and, in the case of SPK, cooperation would be coordinated through RFPK, the organization with the best understanding of the product.

Cooperative Efforts

All of the above named open source software products are the result of cooperative efforts by many groups and individuals, usually working in geographically separate locations, communicating via the Internet. In order for SPK to have the greatest possible impact, it would be highly desirable for developers outside of RFPK to share in the work. We will actively pursue cooperative development by adopting the development methods of the open source community.

Given that the source code will be readily available, anyone with the requisite level of software development skill will be able to make or propose changes to any part of SPK. In the enhanced architecture, however, areas that would be especially attractive for parallel development would be the *client tier* and the *reusable models.*

We plan to provide a number of Model Design Agents (MDA) for the client tier. Our list is by no means exhaustive. Others may want agents that support languages and conventions different from those that we have selected. They may wish to develop MDAs that they believe are more powerful or user-friendlier than ours. In order to encourage the development of MDAs by other groups, RFPK will publish the specifications for the application protocols that comprise the interface between the MDAs and ASPKs.

As described above, SPK will contain a library of reusable models, to which users can contribute. This will be a very important avenue for cooperative efforts.

Commercial Opportunities

Employing an open source license will eliminate the possibility of a commercial entity using SPK source code as the basis of a proprietary product. This does not mean that there will not be commercial opportunities. Indeed, the concept that commercial partners can earn profits while adding value to SPK has been an important aspect of the project since inception. With the three-tier architecture, software firms will be able to develop and to market MDAs which conform to the interface protocol published by RFPK, as long as the products do not contain SPK code. These MDAs might support additional languages not supported by other MDAs, or they might employ different user interface technologies. Another important area of commercial opportunity would be the design of reusable models that would be offered for sale as products or as part of a service.

There would also be an opportunity for *application service providers (ASP)* to offer an SPK service to the research community. (An ASP is a computer service bureau which sells access to specific applications which are run on its computer systems and which the customers access via the Internet.) ASPs typically provide their customers with high quality services while relieving them from having to operate the computer systems themselves.

Relationship of the Three-Tiered Architecture to DML

When RFPK started in 1998, five deliverables were envisioned: SPK, SDPK, SCPK, PCPK and GCPK. Subsequently, based on feedback from potential users and from the Advisory Committee, the five items were replaced by two: SPK and DML. Most of the software development efforts up until now have been directed toward SPK. DML, however, has remained part of the vision. The *Differential Equation Modeling Language (DML)* was to be a component of the *Differential Equation Modeling System (DMS).* Together these exist today as a language definition and a conceptual design.

There is a clear parallel between DML/DMS on the one hand and the Client and Foundation tiers on the other. DML was a model specification language which was to be automatically translated into a driver and linked to the SPK library in a manner conceptually similar to the process by which designs coming from a MDA will be compiled and linked to the SPK library for running on the CSPK.

The difference between these two design approaches is that DML is a new language, unused by any scientist at the present time, whereas the MDAs will be developed to use existing languages. Because reaching users has become a very high priority for RFPK, we do not intend to experiment with new languages at this time. It is conceivable that at some future date, after MDAs for Nonmem, Matlab, and some of the other existing model development languages have been completed, a MDA for DML will be considered.

The Road Ahead

At this time (April, 2003) we believe that we can see the broad outlines of what lies ahead, as we develop the three-tiered architecture. These will be important milestones:

- Transformation of the Software Team to use open source tools and methodologies on a Linux platform.
- Conversion of SPK from Microsoft-only source code to a portable form that can be run on any Unix or Linux system or within the Cygwin environment on Microsoft Windows systems. Retesting of SPK on the Linux platform.
- Replacement of all of the proprietary NAG code within SPK with open source code, so that SPK can be distributed under an open source license.
- Automatic model generation for Nonmen type model designs.
- XML markup language for input to the model generator.
- XML markup language to return results to MDAs.
- MDA that uses the Nonmem language and conventions.
- ASPK that incorporates automatic model generation, queues models for execution on the CSPK and reports status and results back to the web page.
- MDA that transforms results into formats for various analysis and visualization tools.
- CSPK that removes jobs from the queue, executes them and sends results back to the ASPK. The initial version will not be capable of parallel processing.

We estimate that the initial version of the three-tiered architecture will be completed in the second quarter of 2005. During the development period, the core SPK engine will continue to be enhanced with improved numerical algorithms and improved evaluation of results.

After the initial release, enhancements will continue to appear on a regular basis. These will include:

- CSPK capable of parallel computation on symmetric multiprocessors.
- Model library management capabilities for the ASPK.
- CSPK capable of parallel computation on Beowulf clusters of symmetric multiprocessors.
- Additional MDAs.