

All I Really Want... A Wish List for New SAS® Software Enhancements

Peter Parker, US Dept. of Commerce

Abstract

Having used both SAS software and computers for over 21 years, I've seen many changes in both, almost always good. Major improvements will continue to be made and we'll look back five years from now, amused at what we called state-of-the-art today. Remember when a 486 PC with a 20 MHz clock speed, 4 MB of RAM and a 300 MB hard drive was considered the dream machine? Remember when packing a million transistors on a chip was a major breakthrough? The latest computer chips have over 42 million transistors.

In order to advance technology, we must first determine what needs to be improved. While SAS software has proven itself to be adaptable, dependable, and robust, it, too, will become better. Here is my wish list of changes that I hope will become part of the SAS software system soon:

System Enhancements

1. Run- time SAS applications
2. Multi-threaded processing

Application Development Enhancements

1. Import Wizard for most types of data
2. GUI code generator
3. Intelligent Printing
4. Merge Wizard
5. Coding Critique Wizard
6. SAS code templates
7. SAS macro window overhaul

Introduction

I've been using SAS Software for a long time (20+ years) and I've used it on ancient mainframes, keypunching out PROC statements and submitting stacks of fragile punch cards to computer operators. Now I type my code on my recently state-of-the-art PC and submit and review my computing jobs in the privacy of my office. Much has changed since then, quicker and easier submission of number-crunching queries, color-coded code, improved interfaces, and tweaking of the software syntax. However, much has not changed. You still have to write code, use DATA steps and PROC statements, and use logical, correct syntax. There is still Base SAS software that can do anything that lower level languages like FORTRAN and COBOL can do, but with much less typing and debugging. However, if one wants to read directly non-SAS data files like dBase, he either has to write a dynamic data exchange program (DDE, not a coffee break program) or buy another SAS software product like SAS/ACCESS®. A lot has changed, yet a lot has not.

This paper will focus on the backbone product of SAS software, BASE SAS, and how that product should evolve beyond tweaking and more Windows-like graphical user interfaces (GUI). It appears that most of SAS software's growth has been in creating more software products. This progression is desirable for those users who have advanced econometric or graphics needs, but BASE SAS is the only product that most people need for number crunching. They want to perform advanced data manipulations without having to buy the other SAS products like SAS/ACCESS,

which quickly add up the costs of running SAS software.

I've examined the changes that need to be made to the overall architecture of SAS software (system enhancements) and to the programmer interface (application development enhancements). Every year, the SASware Ballot® asks users what changes that they want. This is my formal response. This is my SAS software wish list.

System Enhancements

Run- time SAS applications

Ever since running SAS software on PCs became viable for serious data processing, I've been requesting a run-time version. So have many other SAS programmers. A run-time version of a program is a compiled form of that software that will enable any PC, regardless of whether that software is installed, to use that particular customized application. For example, one can create a Microsoft PowerPoint presentation, save it as a run-time version, and distribute it to other users, regardless if they have the PowerPoint software on their PCs. As long as they have the Windows operating system, they can run that particular PowerPoint presentation. However, they would not have any other PowerPoint capability, such as creating new presentations or editing the current one. Having this similar functionality with SAS software could have many useful applications.

Suppose with SAS software, I create an program that reads a particular SAS data set (e.g., Textile and Apparel Imports for 2001). I can include SAS Macro Windows, so that when the program is run, the user will be able to generate a report, based on parameters that he passes through the screen prompts, such as what particular textiles commodities he wants to see and from what countries they were exported. If one could compile this program in a run-time version of SAS software, then he could copy this runtime program and SAS data sets to a

CD-Rom and distribute it to clients. Then the clients could run pre-made queries on this textile and apparel data, limited only by the scope of that run-time program. They would not be able to run other SAS applications or create other SAS programs. They'd only be able to run that one SAS program that the licensed SAS user has created.

Obviously, the lack of run-time SAS software is legal/business concern, rather than a technological issue with the SAS Institute. Having a run-time version of SAS software may invite license abuse. Rather than having several SAS software licenses within a company, it would need only a few copies for the programmers. The users, themselves, could run runtime queries, instead of relying upon the IT staff. Since there is no run-time software, some companies may buy the next closest thing, SAS/IntrNet™, not only for running SAS software on the Internet, but also for running SAS software on the Intranet. They would have to weigh the cost of SAS/IntrNet software versus individual licenses. Another option around the lack of runtime software is to use an ODBC interface to the SAS system. For example, our office has created applications in Borland Delphi software that interfaces with SAS software to read the SAS data and produce reports.

Multi-threaded processing

Since many servers today have multiple CPUs, it would advantageous to run software that optimally uses all of these processors. In theory, the application will run faster, subject to bottlenecks such as how fast the hard drives can deliver the data to the CPUs to be processed. In order to use all of these CPUs, the software must be multi-threaded.

The SAS software is single-threaded. A thread is a list of instructions to perform a certain task. Multiple-threaded software can perform several tasks simultaneously if the resources are available. Only by running multiple applications such as multiple SAS jobs at the same time could SAS software run faster on a multiple processor system. Each SAS job would not run faster, but they

wouldn't have to share one processor among them.

The Windows NT operating system uses symmetrical multiprocessing where the system can load multi-threaded applications among the different processors, trying to balance the computational load rather than having certain processors always do specific type of processes. Running multiple single-threaded applications could also use these multiple processors.

With SAS version 8 software, there is a new product, SAS/MP CONNECT®, bundled with SAS/CONNECT®. Besides having to buy yet another SAS product, this new product is not a solution for “easy to use” and “true” multiprocessing. Additional SAS/CONNECT coding is required, emulating the concept of logging on to different hosts (i.e., starting a remote SAS session on each processor), where all the hosts are on the same PC. As well, the multi-processing only works when the tasks are independent. One can merge two SAS data sets, but cannot sort the newly combined data set until that merge is complete. The more desirable form of processing, not yet provided by the SAS Institute, is where the two processes are dependent but the former one does not need to be done completely before the latter one can begin. In this example, the SORT process wouldn't need to wait on the merging of two data sets before it can start sorting the combined data. It could start the process as soon as a few records have been merged.

Ideally, the best case of multi-processing involves these three elements:

1. Built into Base SAS software (so one doesn't have to buy yet another SAS product).
2. Works transparently with all existing SAS code (so one doesn't have to modify their current code).
3. Works for all tasks, regardless if independent or dependent.

Application Development Enhancements

Import Wizard for most types of data-

Some people may consider the wide array of software available as being an over abundance of technological riches. We have so many choices and so many ways of doing things. We may have too many choices, leading to confusion. Data can be stored in several ways, spreadsheets (Lotus, Quattro, Excel), rich text, ASCII text, binary, dBase, Paradox, SAS software, SPSS, etc. Often we need to read this information into a SAS program for data manipulation. One can use ODBC interfaces to read directly a dBase file into SAS software or to write a SAS data set into either dBase database files or Quattro spreadsheets. The coding is not difficult. However, for each type of data, we have to write and debug SAS code to handle it. Rather than having to write this code, the SAS software should handle it for you, or at least, attempt it. One should be able to make a LIBNAME reference to a spreadsheet and in the code, write-

```
LIBNAME ABCSPREAD  
c:\mydata\spreadss.wk3;
```

```
.
```

```
.
```

```
DATA XYZdata;  
INSERT abcsread;
```

```
.
```

Here a LIBNAME statement points to a spreadsheet file on the c: drive (c:\mydata\spreadss.wk3) and using an “Insert” command (this keyword doesn't exist, but could be a name for this code), SAS software would figure out how to insert this spreadsheet into the temporary SAS data set (XYZdata). One could have parameters in the “Insert” statement that could specify which ranges of columns and which folders in the spreadsheet to use, and any other relevant information to define how the non-SAS file is to be converted into a SAS file.

In Base SAS version 8 software, there is already a limited import wizard that can read in delimited, comma separated and tab delimited files, all being different types of ASCII text files. One can also read in Microsoft Excel, Microsoft Access, Lotus 123, and dBase files, but he needs to purchase yet another SAS product, SAS/ACCESS. As noted elsewhere, this import/export feature should be bundled with Base SAS software and not be another costly extra.

GUI code generator-

While easy to use, SAS software uses its original paradigm of manually writing code, be it DATA Steps or PROCs or system options. Seasoned SAS users easily can rattle off this code, rarely having to look up the syntax in the many SAS manuals. However, to reuse the popular tirade of the 90's, the SAS Institute needs to have a paradigm shift. An alternative Base SAS software would use objects instead of code to set up a program. However, the programmer would be able to modify the code behind the objects, when necessary. As an analogy, when CompuServe started to provide remote service, users had to memorize keywords and literally type in "Go Mail" or "Go Billing" to navigate around the product. Then AOL 1.0 was released and the user rarely had to type in commands, instead clicking on a "Mail" or a "Billing" icon. Other current software uses this object-oriented look. In Lotus Notes, the information database/email system software, one can create database fields as objects on the screen with a pick list containing all valid responses. All of the Lotus Notes components can be viewed as objects, whether they be database fields, agents, action buttons, forms, sub-forms, views, pages, navigators, outlines, framesets, folders or other Lotus Notes constructs.

Perhaps a SAS-Lite version could be created for neophyte SAS programmers or for experienced programmers to create quick jobs. There could be SAS objects for SAS data sets, non-SAS data sets, and SAS work files, and they all can be dropped into action boxes that filter, sort, and merge these objects. The results of these actions boxes can be connected to generator boxes

that print reports, create frequency tables, run regressions and other SAS PROC features. A programmer would have access to an alternative view where he can see the code generated and then change it, if desired. Just like Netscape Composer may create a web page, one could open the generated html file with any ASCII editor and make changes to it. As improvements are made to the SAS-Lite interface, the programmer will have less reason to edit the actual code.

Since pictures are more intuitive than words, one can look at a page of objects and understand what a program does, rather than having to read pages of code. This concept is similar to creating a program flowchart that is also the program.

Intelligent Printing

Printing is a problem for all software, including SAS software, spreadsheets, databases and word processing. Printing should be as effortless as pressing the print button and what's on the screen will be transferred perfectly to paper. It doesn't always work that way. Often, especially in SAS software, you have to adjust the fonts and page sizes and play with the code to have the report fit correctly on the pages. With some fine-tuning, page breaks will be correct, instead of several lines later on the next page. It becomes an art to have the printer output suitable for distribution.

One solution suggested to me years ago by a software representative (not from the SAS Institute) is to go completely paperless. The solution to printing is not to print. However, in many ways, paper is better than a computer screen. It has better resolution. It can be marked up by hand and it is more portable; you can fold it up and put in your back pocket. I'm always annotating it, underlining words and highlighting sections. Shifting paradigms here is not a recommended alternative.

The solution is to have the software, operating system and printer all communicate better, with minimal user intervention necessary. The only intervention suitable would be to tell the printer to duplex (print on both sides of the

paper) or to decide which printer to use. All other housekeeping duties should be done by the computer, regardless of the fonts used or of what type of report or table is being used. One should be able to use SAS options to determine page-size or line-size, but I doubt many programmers would bother if the software took care of it well. I only change these options from the default when my reports don't print correctly.

Merge Wizard-

Perhaps the most confusing part of Base SAS software is using the merge feature. While it has the powerful function of combining data sets together, it may cause problems not discernable immediately to the programmer. For instance, does he want to merge two data sets only if they have common values for a certain variable? Does he want multiple records when there's a match or just one? What if the data sets have common variables besides the matching one? What value will the variable then have, after the merge? It all depends on how you define "merge". A graphical user interface (GUI)-based Merge wizard might ask the right questions to the programmer to determine what he really wants. The merge wizard could illustrate the different types of merges, whether it's one-to-one merging or match merging and then show the results, using simple examples of combining two small data sets.

This merge wizard could evolve into a more advanced assistance to programmers. If you just want to append one database to another, the wizard may suggest that you simply use a "SET" statement or some other technique that is better suited. Eventually, one won't even need the many SAS software manuals. One can have assistance via the SAS wizard. The merge wizard could become an artificial intelligence agent for all SAS software features.

Coding Critique Wizard-

It's not hard to write adequate code. You sketch out the basic workflow logic, type the

code as you think it up, create some test data, debug the program on the test data, and then run it against real data. If it runs within a reasonable amount of time, the program is a success. The job is done and it's time to move on to other projects. Never mind that the program may have some wasteful sorts or inefficiently written DATA steps. It may not matter at the time that the code is so poorly written that a year from now someone may have to spend long hours attempting to decipher it.

Are you writing good SAS code? Is this SAS code that you would want someone else to read? Would you publish it in a SUGI paper? We shouldn't ask only whether it works and what can we get away with. While shame does have its place in programming, among other aspects in life, your current code also may not work in the future. I found that some of my SAS 6.12 code did not work in SAS 8.0. I had to tighten up the code. That means I had to be more careful with coding syntax. The message is basic. Do it right the first time, especially if you plan to reuse it.

Artificial Intelligence software that reads your program and critiques it may be a unique approach to fixing clunky programs while they're still fresh. A similar idea is used in checking out websites. You submit your website address to one of these services and it rates your page, based on load time, browser compatibility, dead links, spelling and HTML design. A SAS code critique wizard could give warnings on merges, inform you of inefficient coding, and give examples of better coding. For instance, if you don't have a KEEP statement and you don't use all of the data fields, it should remind you to use a KEEP statement in a DATA step. If you have included a KEEP statement but not use all of the fields listed in it, the wizard should inform you to trim down that KEEP statement.

SAS code templates-

I'm always taking old programs and recycling them. Why write a program from

For example, Lotus Notes, the information database and workflow software, provides several database templates as a starting point. They have templates for creating a document library, a registration interface, and a discussion database, among others. The SAS Institute should be able to implement similar templates. For instance, you may have a template that reads in SAS data sets, filters the records, and run reports based on parameters passed through macro windows. Another SAS template may read in data, run validity checks and then run certain statistical procedures. Another SAS template may be set up to read in non-SAS data sets, filter out the data, compute new data fields and then create html output for a web site. The SAS Institute could determine, with market research, how most people use SAS software, and create these templates based on the most common needs.

SAS macro windows provide an interface for users to pass parameters through simple windows forms. If one wants to query the textile and apparel import data for knit shirt imports from China for the year 2000, one may create a SAS report, by querying the data using a simple macro window that may look like this:

However, this interface is only a “simple” window, not much different than the clunky front ends to DOS software that we stopped using years ago. It needs more component objects, such as check boxes, radio boxes and dialog lists, to make it easier to use. In the above example, one needs to type in the first two parameters. For the type of commodity, is it spelled “knit shirts” or “knit-shirts”? Is the country “China” or “People’s Republic of China”? What if it’s for a country that hard to spell like “Madagascar” or has changed its name several times, like “Burma/Myanmar”? What if you have 20 years of data? How do you get it all on the form? What if you select more than one year? Can you have multiple commodities and multiple countries? Obviously this macro window has its limitations. Here is an improved window:

In order to be a “complete” interface, it needs most of the work done by the computer, minimizing errors such as the user misspelling the country name or typing in “swimsuits” when “swimwear” is the appropriate term used in the database for beach clothing. Instead of a “DOS” approach, SAS software needs to emulate the interface that Lotus Notes and Borland

Delphi enables application developers to create.

Conclusions

There's always room for improvement.

But before you can make improvements, you need to determine what needs to be changed. You need a wish list. Most of the changes listed here are for repairing existing SAS software products (macro windows, intelligent printing). Some are major systems changes, such as making SAS software a true multi-threading application. Some are major transitions, such as replacing written code with objects/symbols. One should expect many of these changes within the next five years, especially the small fix-ups. However, I don't anticipate that the major changes may ever occur. SAS software has been around for over 25 years, and it has a responsibility to support legacy systems. SAS software versions 9, 10, and 11 will still need to support SAS version 5 and 6 applications. While that may be the responsible action to take, it also limits how much the software can evolve. Backward compatibility has that inherent short-coming, just as Microsoft still has some ties to running DOS applications, although its Windows and Windows 2000 operating system are far beyond the simple command line operating systems.

Will SAS software evolve like I've outlined in this paper? Not likely, but after all, this is only a wish list.

References

Bentley, John E. (2001) "SAS® Multi-Process Connect: What, When, Where, How, and Why" in SUGI26 Conference Proceedings. Cary, NC: SAS Institute.

Garner, Cheryl (2000) "Multiprocessing with Version 8 of the SAS® System" in SUGI25 Conference Proceedings. Cary, NC: SAS Institute

Parker, Peter (2000) "Optimizing SAS® Software on Windows NT: A Guide to Performance Tuning Your Applications

Server" in SUGI25 Conference Proceedings. Cary, NC: SAS Institute

Parker, Peter (2000) "SAS® Software Macros: You're Only Limited By Your Imagination" in SESUG Conference Proceedings and NESUG Conference Proceedings. Cary, NC: SAS Institute

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Peter Parker
US Dept. of Commerce
Room 3100
1401 Constitution Ave., NW
Washington D.C. 20230
202-482-1449
peter_parker@ita.doc.gov
<http://otexa.ita.doc.gov>