SESUG Speaker Sharing Program

To arrange for a SESUG speaker, contact Marje Fecht at Marje.Fecht@prowerk.com

Speaker:

Frank DiIorio
CodeCrafters , Inc.

Bio:

Frank DiIorio is President of CodeCrafters, Inc. and the author of "SAS Applications Programming: A Gentle Introduction" and (with Ken Hardy) "Quick Start to Data Analysis with SAS." Frank has been active in the SouthEast SAS Users Group (SESUG) since its inception, co-chairing the 1994 and 1996 conferences.

He has, much to his astonishment after doing the math, over a quarter century experience with SAS software. His new book, "The Elements of SAS Programming Style," (working title) will be published "real soon now."

When not writing *about* SAS, Frank writes *in* SAS, primarily data management and reporting applications in the pharmaceutical industry. A native New Yorker, he has lived in Chapel Hill, North Carolina since 1974 and sort of buys into its claim of being the "Southern Part of Heaven."

Presentation Topics:

- The Elements of SAS Programming Style

- The SAS Debugging Primer

- Dictionary Tables: Essential Tools for Serious Applications

- Writing the "Best" Program: The How and When of Efficient Programming

- Program Comprehension: A Strategy for the Bewildered

- Rules for Tools - The SAS Utility Primer

SESUG Speaker Sharing Program

Abstracts:

The Elements of SAS Programming Style

The generalized nature of SAS software almost guarantees that "n" users will develop "n" unique solutions to even basic tasks. The gap between the task correctly performed by the programs and the disparate code is, for the most part, due to programming style. This presentation discusses a set of generalized programming style guidelines useful to both experienced and novice SAS programmers.

It first investigates general principles of program design, those aspects of the analysis and coding process common to all aspects of SAS programming. The next sections focus on coding guidelines for the DATA step and procedures. Finally, debugging techniques are addressed.

The presentation contends that "good" programming style usually results in programs which are more effective in terms of both human and machine resources. The intent is not to pronounce one style good and another lacking, but to simply outline an experienced user's guidelines and gently prod other users to examine their programming habits. These habits will become critical to the success of organizations as SAS software becomes embedded in more environments and organizations.

The SAS Debugging Primer

Meet an accomplished SAS programmer and you meet someone who's probably learned by making (and fixing) lots of mistakes along the way. The breadth of the SAS System's target applications, the variety of its "dialects" (Base SAS, macro, SCL, IML, SQL), and the quirky procedural/non-procedural environmental mix conspire to make mastery of the SAS System a slippery slope to ascend. Debugging is the art of gracefully recovering and learning from falls during the ascent.

This paper discusses techniques for debugging SAS programs. Its purpose is two-fold. First, it provides behavioral and technical tips for fixing code (how to read error messages in the SAS Log, knowing when there is a problem with the program even if SAS says there isn't, using the DATA step debugger, identifying system options, using PROCs for data validation, using macro variables to control debugging output, etc.) The second focus of the paper is its presentation of design and coding methods that make the programming process more reliable, thus reducing the need for debugging in the first place.

The paper's target audience is relative newcomers to the SAS System. More seasoned users may find or rediscover some of the techniques and features being discussed. Emphasis is placed on Base SAS and the macro language, although the techniques themselves are applicable to SCL and other products.

Last Update: 5Oct2004

Dictionary Tables: Essential Tools for Serious Applications

Dictionary tables and views are some of the most useful and least heralded tools in the SAS System. They contain a wealth of information about SAS datasets, catalogs, system options, and external files. While much of this information is available via cumbersome procedure output datasets, dictionary tables and views (which we will simply refer to as "the tables") put this information at the disposal of the application developer in a straightforward manner. The use of dictionary tables in combination with the macro language and PROC SQL, two much more utilized SAS products, provides a powerful set of tools for the SAS programmer.

This paper: explains the organization of the tables, describes their content, and presents several short applications demonstrating how the tables can be used in combination with the macro language and PROC SQL to create more efficient and dynamic applications. The reader should come away from the discussion with an understanding of the benefits of the dictionary tables as well as some ideas for how they may be used in their programming environments.

Writing the "Best" Program: The How and When of Efficient Programming

It's relatively easy to write programs that optimize the use of CPU and other machine resources. There is a large and continually growing body of literature on the subject. What isn't as straightforward is knowing **when** to employ the techniques - blind implementation of tuning techniques is often not required by the task at hand and can sometimes even be counterproductive.

This paper addresses both the "how to" and "when to" aspects of writing efficient programs. It describes design and coding techniques that conserve hardware resource usage. It also identifies other, non-machine implications of their usage that could dissuade the programmer from their use. For example, using temporary array elements is more efficient than using named elements but has the documented-but-obscure behavior of retaining values across observations. Maintenance of such code by other than "seasoned" and up to date programmers can be unexpectedly problematic.

The concept of efficiency used in the paper includes all aspects of the program life cycle. We apply the "how and when" question to system design issues, system startup, DATA steps, procedures, and macros. Emphasis is on Base SAS software. The reader should finish the paper comfortable with the idea that the "best" program is not always the one that minimizes hardware resources.

Last Update: 5Oct2004

Program Comprehension: A Strategy for the Bewildered

Here, unfortunately, is a not uncommon workplace scenario. A neophyte SAS programmer is assigned to maintain, debug, or enhance an application. The atmosphere is sink or swim, the system is complex, the code is sophisticated, the documentation is scant, and the programmer is bewildered. Questions slowly take shape. "What, exactly, am I supposed to do?" "What part(s) of the application need my attention?" "Will changing program X affect program Y?" And, most critically, "Where do I start?"

What the programmer needs is a strategy for understanding the program, then finding its "sweet spots" as efficiently as possible.

This paper presents a generalized approach for programmers, particularly SAS "newbies", to develop an understanding of how applications work. It also shows how to translate this comprehension into effective coding. The paper identifies and discusses the rationale for questions the programmer should ask about: task definition, program-level code, supporting code, system design and specification documents, and required domain knowledge.

Attendees should come away with a better understanding of how to frame the programming problem and effectively gather the resources needed to obtain a solution. They will also come to believe that the coding of, say, a DATA step is usually simple, but the real art of programming is learning what to code, and why.

Rules for Tools - The SAS Utility Primer

Let's start with the premise that good programmers are lazy by nature. They want to use tools such as formats and ODS for execution-time efficiency or to pretty-up our output, functions to perform calculations, and so on. Another hallmark of a good programmer is a keen eye for pattern recognition. Rather than rewrite basically the same program over and over, they identify similarities and parameterize the program, making it into a general-purpose program, a "utility."

This paper steps through the life cycle of a simple utility. It starts with "naïve" code that doesn't exploit program similarities, then illustrates how a general-purpose utility may be developed. It ends with the initial program becoming a call to a simple, powerful routine in a macro library. The transition from simple, brute-force programming into a compact, general- purpose utility isn't a random event. The last sections of the paper present a set of design principles for utilities.

Although we focus on Base SAS in Version 9.0, the principles and techniques are readily extended across SAS versions and products. The reader will come away from this paper with an appreciation of both the process and the tool set required to build generalized programs.