developer.*
The Independent Magazine for Software Professionals

Syndromes of Forgotten Programmers

by Kevin Cauble

Editor's note: This essay originally appeared in the August, 1991 issue of Software Maintenance News, *which was published by founder and Editor-in-Chief Nicholas Zvegintzov between 1983 and 1994.*

Nothing in my Computer Science curriculum prepared me for life as a maintenance programmer.

In my first tour of duty, I inherited a mixed bag of programs. Some were written in-house over several decades. Others were written by consultants who had disappeared. Some had been obtained from other shops and adapted (as little as possible) to our needs. Most had withstood a Honeywell-to-IBM conversion. Others were further subjected to a VSAM-to-SQL conversion.

As I approach the four-year mark in my career, and before I become a jaded professional, I thought I would document some accepted programming practices my Computer Science professors forgot to teach me.

The Rube Goldberg Syndrome. Rube Goldberg was a cartoonist famous for incredibly complex solutions to life's simpler problems. His programming counterparts never tire of demonstrating to everyone what incredibly complex and talented programmers they are...even in simple situations that don't call for it.

The Mother Hen Syndrome. Mother Hens have a bad case of nesting instinct, at least when it comes to `IF..THEN..ELSE` constructs. I recently had to debug a program with 14 levels of nested `IF`s. These probably do not occur as often as they seem. It's just that if over-nested `IF`s exist in a program a debugging trail generally leads to them.

The Garbage Collector Syndrome. Garbage Collectors are incapable of deleting a line of code, no matter how useless or obsolete it might be. The programs they work on resemble mazes, full of commented-out code and unused modules. Imagine what these programs will be like in another 10-20 years!

The Cryptographer Syndrome. The Cryptographer refuses to take advantage of one of COBOL's few advantages—30-character variable and paragraph names—which makes possible descriptive and meaningful names. So why name a variable `XYZ`? Or name a paragraph `725-DO-IT`? These programmers probably spend the programming time they save working cryptographs. While maintenance programmers slave away on overtime trying to decipher the Cryptographer's code.

The New Math Syndrome. Or I assume it must be New Math, since we didn't count that way when I was in school.... I thought the purpose of using numbered prefixes for program modules was to make them easier to identify and locate. But the New Math programmer is quite comfortable having program module `250-` precede module `200-`, which is after module `900-`, etc. Or how about 32 modules, all prefixed `400-`?

The Also Known As Syndrome. This syndrome is evidenced by single data elements referenced under a variety of names. There is no greater joy for a maintenance programmer than thinking you've located the last reference to a variable, only to find it is moved to a new name and the tracking process begins anew.

The Clone Syndrome. Cloners have never written a program from scratch. They find an existing program that somewhat resembles the current assignment and change only as much code as needed to make it work. If you follow this method, be sure *not* to change any variable or paragraph names to reflect the new situation, and do *not* delete any unused code. That would make life too easy for maintenance programmers.

The Railroad Engineer Syndrome. The Railroad Engineer is switch happy. Instead of taking the time to understand a program's logic and weave in a solution, the Railroad Engineer's answer is another special purpose program switch. But for every n switches, there are 2^n possible combinations of switch settings to boggle the minds of maintenance programmers.

The Author Unknown Syndrome. When I first found programs with no author listed, I thought I had found Gerald Weinberg's "Egoless Programmers." After debugging and amending many of these anonymous programs I understand the real reason. I'd be ashamed to associate my name with programs like these also.

Count me in favor of adding a course in software maintenance to Computer Science curriculums. I envision a course where professors hand out programs written by other students in previous semesters. The student's task is to make modification to the program or add a new function. Learning to read another programmer's code was the biggest ingredient missing from my preparation for the real world.

###

While driving a Red Cross Bloodmobile, Kevin Cauble got bored and joined a local university to take history classes. Unable to get the classes he wanted, he took a computer class as a lark. After the first minute of the first class, Kevin knew what he wanted to do. He graduated with a degree in Computer Science and is currently working as a Programmer/Analyst at a small liberal arts college in North Carolina. Kevin can be reached through the editorial staff of **developer.***.