D3: Sacred software cows: linguistic quibbles

Les Hatton, lesh@oakcomp.co.uk

Dec 1995

Last month I promised to put the boot into a few beliefs which we have held dear for years and which are not supported by software systems measurements and I promised a few surprises. I'll start with languages. Faced with software faults, the first piece of advice one gets offered is to switch to the new language of the moment - e.g. C++, VizBaz etc. The computer programming language industry is indeed a mighty one. When the U.S Department of Defence standardised on the Ada language in the '80s, it did so because defence contracts were being implemented in literally hundreds of programming languages. Since then, Ada has frequently been spoken of as the "language of choice" for safety-related systems and is widely thought to produce more reliable systems because of its inherently superior features. In contrast, in "serious" circles, the mere mention of the language C for example, can lead to bones being pointed at you. Other languages invite various comments, eulogistic from their supporters and disparaging from their detractors. So, what does the data tell us ?

The simple truth which emerges from many studies of failure rates in non-object orientated software suggest that we make about the same number of faults per line (i.e. fault density) whatever language we use. C, Ada, Fortran, assembler, Pascal - they are all about the same, so it doesn't really matter, provided you don't use assembler which takes a lot more lines to achieve anything. In other words, all languages are stricken to about the same extent and indeed, some of the most reliable systems ever written are in C. So much for intuition. You may notice that I excluded object-oriented software in the above, as there has been little data until recently. The object-oriented (OO) school by dint of massive advertising, deep thought and no measurement are trying to convince the software development world that this is the only way of producing good, re-usable, portable, reliable software. As a result, the take-up of languages such as C++, (a.k.a. C+- by its detractors), is little short of phenomenal. So is OO really the way to go ? Er, not when it comes to reliability, it isn't. Some very recent data on C++ systems suggests that the fault density is very similar if not slightly worse than equivalent projects in C. In other words, it is probably a great leap sideways, like so many of our non measurement based leaps.

So where do we go from here ? One of the big problems is that programming languages keep growing to keep market share, whilst reliability and other important intrinsic properties require the use of safe, well-defined subsets of the language only. We will make progress when the latter becomes more important than the former, driven probably by software defects in the next generation of consumer electronic products. See you next month.