# D3: Sacred software cows: small components are beautiful

Les Hatton, lesh@oakcomp.co.uk

Mar 1996

I promised last month to discuss the notion that decomposing a design into small easily understood components makes a more reliable system. First of all, hands up if you believe this ... Well, if you do, then you are in an enormous majority. This after all, has been one of the foundation stones of software engineering for the last 30 years, although of course we disguise it by giving it a nice complicated name such as structured system decomposition or somesuch. This advice is repeated in all training courses of which I am aware and is repeated to all students of the subject. - "Break your system down into small simple components and this leads to the most reliable system". It is also repeated in software engineering standards, and is therefore thoroughly and safely entrenched in all software development, high-integrity or otherwise. Some companies even limit components to no more than a page, a component being a subroutine or function in this sense.

I also believed this myself for many years, although I had never seen any evidence to support it. It seemed so self-evidently obvious, that you wouldn't even begin to question it. Well it's wrong, or at least overwhelmingly likely to be so if I put my statistician's hat on. What basis can I possibly have for making a statement like this ? The answer is of course data. During the last ten years, evidence has been slowly building until it can be ignored no longer. All this evidence shows the same thing: "In any real system, the smaller components contain proportionately far more faults than the larger ones". The nature of the published evidence is astounding. It covers systems written in Assembler, C, C++, Fortran, Ada, and Pascal, and spans scientific subroutine libraries, giant operating systems, communications systems and all manner of others. In other words, they have little if anything in common, other than that they are all software systems. The data also strongly suggests that there is a bath-tub shape to the defect density curve (defects per 1000 lines of code) when plotted against component size. The smallest components have high defect density; medium sized components have the lowest defect density; and to restore the intuitive view, very large components, (greater than around 400 lines or so), have high defect densities again. The lowest defect-density occurs at about 200 lines, and is surprisingly insensitive to the programming language used.

Oh dear, where on earth do we go from here ? Next week, I'll discuss some of the possible reasons for this. Whatever the reason is, we certainly can't ignore the data any longer. Its time for some serious rethinking in software design.

January's Mac errors. Not a good month ! - 66.5 hours use; 17 defects of which 6 led to a reboot and 1 to a re-format !