D3: Small components AREN'T beautiful

Les Hatton, lesh@oakcomp.co.uk

Apr 1996

Last month, I quoted a very well-established fact from the analysis of numerous diverse software systems that "In any such system, the smaller components contain proportionately far more defects than the larger ones". This is also true of very large components, bigger than around 400 lines or so, leaving the medium-sized ones (around 200 lines) as (by far) the most reliable. If this weren't enough, the phenomenon is apparently independent of programming language with languages as diverse as Assembler and Ada showing strong quantitative and qualitative similarity. Finally, I admitted that this completely contradicts the intuitive view and may well disturb many of you - it disturbed the hell out of me when I found I could ignore it no longer.

This month, I thought I would try and explain why. The following suggestions have all been made to attempt to explain this.

- The more components there are, the more interfaces there are and therefore the more defects there are. If this is true, the nature of the defects is subtle because the studies include languages like Ada which guarantee interface consistency and languages like Assembler, which do not.
- Programmers take more care over bigger modules than smaller modules. Well they ought to take even more care with very big modules, but these are also very unreliable like the smallest components, so this doesn't ring true.
- Not as much functionality is used in bigger modules so they don't fail as often. This doesn't explain why defect density has a minimum with medium-sized components and it also has the same problem as the previous suggestion vis a vis very large components.

My own belief is that the language independent nature of the phenomenon suggests strongly that it is a natural side-effect of the way we perform symbolic manipulations and I have recently published a mathematical model based on the physiology of the human memory system which predicts the logarithmic / quadratic defect density behaviour observed in real software systems.

However, these are all hypotheses open for refutation or further support, and whatever the explanation, the data won't go away, and we will have to live with it. For reasons which I will go into next month, these observations also prejudice the notion that concepts we currently hold dear such as software reuse and object orientation will lead to better, more reliable systems. In other

words, most of our conceptual bases of software development are at risk to some degree. Where do we go from here ? Well, to know is half the problem. Now we have to sort out the implications for software design, and also to accept that some defects may be inevitable.

February Mac errors. Another poor month ! - 71.5 hours use; 24 defects of which 13 led to a reboot ! Joint lemons of the month were Microsoft Mail and Apple Remote Access with 6 each. At least there were no re-formats this month.