D3: Small components AREN'T beautiful, continued ...

Les Hatton, lesh@oakcomp.co.uk

May 1996

Last month, I attempted to explain the result of a number of important recent experimental studies which showed that in any system, the "medium-sized" components are proportionately much more reliable than either the "smaller" components or the "larger" ones. In this sense, "smaller" meant up to around 100 lines or so, "medium" meant somewhere between 100 and around 3-400 lines and "larger" meant larger than this. Even more surprisingly, the effect is apparently relatively independent of programming language. Although there are several possible models for such behaviour, the data seems inescapable and palpably conflicts with the intuitive view.

One particular aspect of this result which I will pursue here, owing to its profound implications, is the logarithmic behaviour of fault occurrence with size, observed in both "smaller" and "medium" components in any system. This alone turns three cherished beliefs completely on their head.

Belief 1: If you have 1000 lines worth of 'functionality' to implement, it is best to implement this as say 100 x 10-line components because these will be very reliable owing to their small size. Almost certainly WRONG. The logarithmic behaviour of error with size suggests that implementing the system with 10 x 100-line components will be much more reliable, (100 log 10 ¿¿ 10 log 100).

Belief 2: Re-use has to improve matters. NOT NECESSARILY. First, distinguish between two kinds of re-use. The first kind is re-use within the same system to modularise functionality and hopefully reduce its overall size. This is one role of perfective maintenance. In this case, the logarithmic dependence predicts that unless the overall size reduces dramatically, (corresponding to at least 70% re-use), the system may get worse before it gets better. The second kind of re-use is when you re-use components in a separate system. Here, if the reliability of the re-used component is less than the average reliability of the new system, (not uncommon), things again get worse. It has been frequently observed that re-use fails to deliver expected benefits.

Belief 3: Object-orientation is so coruscatingly wonderful, that it is pointless even to question just how wonderful it is, and the collection of any data is therefore irrelevant. Readers of this column will know that I don't wholeheartedly subscribe to this Hegelian view. In fact, I believe that it will become

one of the greatest own goals yet scored by the computer industry's aversion to the scientific method. The logarithmic behaviour of fault with component size suggests that breaking a system up into vast numbers of small components is exactly the WRONG thing to do. Furthermore, data is beginning to emerge which substantiates this view as I will discuss in more detail next month. See you then.

March Mac errors. Lots of holidays this month so only 24 hours use, which still managed to yield 12 defects of which 3 led to a reboot. Lemon of the month was Now Calendar v. 3.5 with 5.