D3: Sacred software cows: maintenance

Les Hatton, lesh@oakcomp.co.uk

Jan 1996

I'll talk this month about software "maintenance". One of the main problems which inhibits improvement of quality and reduction of cost in many IT companies is that they don't really know what IT costs them, so progress is random at best - "If you don't where you are, a map won't help", as the estimable U.S. software process guru Watts Humphrey reminds us. However, since Barry Boehm's pioneering book on software engineering economics appeared a decade ago, we now know a great deal about software costs, so let's do a few sums.

For example, this source and numerous others support the notion that in the overall life-cycle of a given piece of software, only around 20-30% of the total money invested is spent on the development itself. The other 70-80% is spent on what we euphemistically call maintenance, which is split into three categories: corrective - fixing bugs, adaptive - changing things because we changed our mind about functionality and, perfective - improving things without changing the functionality. It was once thought that corrective maintenance only comprised around 20% of this, but prestigious studies from NASA amongst others, now suggest it is nearer 50%. So, for every 5 pounds we spend on software, we spend 1 pound developing it, about 2 pounds correcting it and another 2 pounds fiddling with it. In other words, fixing bugs after "release" costs about twice as much as the whole of the development did. I guess you may think I keep rattling on about reliability because of my professional interests in software safety. I do, but as you can see here, unreliability costs a huge amount of money in both non safety-related and safety-related software, so reliability is going to be an important strategic issue in the next few years. The so-called maintenance crisis is actually one of reliability.

So why do we attach so much importance to development and so little to testing ? The answer is because development is supposed to be interesting and creative whereas testing is supposed to be dull and unimaginative. This is utter piffle and companies which ignore the above cost distribution do so at their financial peril as software development becomes a global industry. Witness for example the massive growth in sub-contracted software development in India.

So maintenance is a figment of our collective imagination. We don't develop, release and maintain it at all. Instead, from its earliest days until it expires, it is in a state of continuous development, with periodic releases to the end-user.

To close, I mentioned in an earlier article that I have been logging errors on my Macintosh and various bits of software since July, so from now on I'll give you a running defect report for the last complete month before I write each article. November: only 29 hours use (I was away in Japan); 15 defects of which 4 led to a reboot. Sigh.