D3: Distributed Blame Systems ...

Les Hatton, lesh@oakcomp.co.uk

Aug 1996

Last month I expressed a desire to know a little more about the Ariane 5 self-destruction which spread a very large amount of European taxpayers money in very tiny pieces around the mangrove swamps and savannah of Kourou on June 4. The European Commission responded reassuringly quickly with an independent enquiry and all is now revealed. It was a classic bug, whereby a 64 bit floating point number was stored in a 16 bit integer using the language Ada. Now Ada lovers will glowingly tell you that the strong typing mechanism of Ada prevents this sort of numerical vandalism unlike the merry world of C and C++ for example. However, what the same Ada lovers won't tell you is that Ada explicitly provides a way of deliberately breaking the rules, called unchecked conversions. Not only that, but when I talked with a number of major Ada users in 1994-1995 when writing a book on high-integrity systems development, I was told that the rules are regularly broken in practice. In other words, Ada is not a strongly-typed language at all. In a true strongly typed language, such vandalism is impossible. In Ariane 5, 7 variables were subject to such conversions but some of them were unchecked because they "couldn't cause a problem". This was true in Ariane 4 from where the software originated, but was not true in Ariane 5, which had a different launch trajectory. So, once again, like Therac-25, blind software re-use explicitly led to a very serious failure. Not only that but as the same software was used in both channels, both channels failed simultaneously. It is finally worth noting that a very similar problem led to the Shuttle Challenger missing the Hughes F/6 Intelsat satellite in space in 1992, and that many high-integrity standards specifically forbid this practice in whatever language is being used, (it is detectable before compilation).

The independent enquiry came to several conclusions but did not seek to lay blame, emphasising instead the distributed nature of the development, (and therefore presumably of the blame). Amongst 13 other recommendations, it recommended that critical software items should be under Change and Configuration Control, implying that they are not currently. If this is the case, this deficiency alone would lead to a level 1 (i.e. chaotic) rating in the Carnegie-Mellon software process maturity model. Finally, commenting on inadequate test procedures, it noted that software should be assumed to be faulty until proved otherwise, rather than the other way round.

So once again, we see a well-known bug occurring again when it shouldn't, and we see that software engineers habitually assume that software is OK until proven otherwise, rather than the other way round, which reinforces once again the wisdom of learning from our mistakes. Word 7 in Windows 95 thinks there are 493 words in this article but Word 5.1 on a Mac thinks there are 505. I think they are both wrong. What a wonderful Word we live with.