Please wait or please slit your throat

As I have written at length on over the years, the systematic production of computer programs without defects is a figment of the imagination. As a result, every programmer has two obligations. The first is that when the system they are building fails, (and it will), it should fail in a way which has the least impact on its users. Secondly, when it fails, it should provide enough information about the failure that its cause can be identified and corrected quickly to avoid future re-occurrence. Anything less is simply poor engineering.

Although both of these are design issues, the first is intimately associated with the nature of the system itself. For example, modern fly by wire fighters have the aerodynamic properties of a dishwasher quite deliberately for manoeuvrability so if a flight control software system fails, it may not be appropriate to hand control to the pilot because they can't fly it either. Failures to provide the second of these is incompetence and yet providing sensible error messages or even an error message at all seems beyond many programmers still and I have a fine collection of idiotic error messages issued over the years. Here are a few of my favourites:-

“System over-stressed”, issued by a cash-register system of several cash registers networked together with a printer in my local pub. This was the programmer's attempt to communicate to the outside world that the printer had no paper.

“Button push ignored”, issued by the flight management system of a MD-11 aircraft. Just imagine, a pilot presses a button and the computer system says “Shan't”. The unfortunate pilot also noted that the manuals for the aeroplane's software systems “appeared to have been written as if by creatures from another planet.” Don't you love it.

“... failed for the following reason: the operation completed successfully” - issued by the ever inscrutable Service Control Manager in Microsoft Exchange.

“Please wait ...”, issued by the flight management system of an Airbus A340, shortly after its entire instrumentation went blank in an emergency approach to London Heathrow in 1994.

“More than 64 TCP/IP or UDP messages ...”: this turns out to be “Modem not switched on”, when translated into the language of William Shakespeare.

Occasionally, the programmer gets into the swing of things and issues a poignant plea such as the following magnificently idiosyncratic example from a Macintosh G3:- “Server not responding ... The network failed (what this means is 'to be supplied in a future version of the MacTCP documentation'; fat lot of good that does) (tcp:116)”

You are probably getting the feeling that I'm a little critical of the communication skills of many programmers, which brings me nicely to what sparked all this off. My university uses Microsoft Outlook web access for e-mail. This has a built in quota system which e-mails you from time to time when you overflow. Last week I sent a whole load of important e-mails. Several hours later, one contact said he had not received anything so I checked to find that none had been sent and all were lost. In other words, when the limit was exceeded, the programmer's response was to issue .... nothing, not a sausage. I suppose “Mail-box full, message not sent; save ?” was a little too much to expect. Doh.

l.hatton@kingston.ac.uk